Two Comments on Targeted Canonical Derandomizers

Oded Goldreich

August 18, 2019

Abstract

We revisit the notion of a *targeted canonical derandomizer*, introduced in our prior work (*ECCC*, TR10-135) as a uniform notion of a pseudorandom generator that suffices for yielding $\mathcal{BPP} = \mathcal{P}$. The original notion was derived (as a variant of the standard notion of a canonical derandomizer) by providing both the distinguisher and the generator with the same auxiliary-input. Here we take one step further and consider pseudorandom generators that fool a single circuit that is given to both (the distinguisher and the generator) as auxiliary input. Building on the aforementioned prior work, we show that such pseudorandom generators of constant seed length exist if and only if $\mathcal{BPP} = \mathcal{P}$, which means that they exist if and only if the previously defined targeted canonical derandomizers (of exponential stretch, as in the prior work) exist. We also relate such targeted canonical derandomizer to targeted hitters, which are the analogous canonical derandomizers for $\mathcal{RP}$.

An early version of this work appeared as TR11-047 of *ECCC*. The current revision is quite minimal and cosmetic in nature.

1 Introduction

In prior work [4], we presented two results that relate the existence of certain pseudorandom generators to certain derandomizations of the class $\mathcal{BPP}$. The first result referred to the standard notion of a uniform canonical derandomizer (as introduced in [5]) and asserted that such pseudorandom generators of exponential stretch exist if and only if $\mathcal{BPP}$ is effectively in $\mathcal{P}$ (in the sense that it is infeasible to find an input on which the polynomial-time derandomized algorithm errs).[1]

The second result referred to a new notion of a canonical derandomizer, which was introduced in [4] and called a *targeted canonical derandomizer.* This notion is the subject of the current note. We mention that it was shown in [4] that targeted canonical derandomizers (of exponential stretch) exist if and only if $\mathcal{BPP} = \mathcal{P}$.

The foregoing notion of a targeted canonical derandomizer was derived as a variant of the standard notion of a canonical derandomizer, which is required to produce sequences that look random to any (linear size) non-uniform circuit. Specifically, a targeted canonical derandomizer is only required to fool uniform (deterministic) linear-time algorithms that obtain any auxiliary input (of linear length), but the generator is given the same auxiliary input. (This auxiliary input represent the main input given to a generic probabilistic polynomial-time algorithm that we wish to derandomize.)

In this note we revisit the notion of a targeted canonical derandomizer. Specifically, we take this approach to its logical conclusion, and consider pseudorandom generators that fool a single circuit that is given to them as auxiliary input. (This circuit represents the combination of the probabilistic polynomial-time algorithm that we wish to derandomize coupled with the main input given to that algorithm. In terms of the notion of a targeted canonical derandomizer as defined in [4], we replace the free choice of a uniform (deterministic) linear-time algorithm by a fixed choice of an evaluation algorithm for circuits.)

We stress that constructing such generators is not trivial. In fact, building on the ideas of [4], we show that *such pseudorandom generators* (of logarithmic seed length (equiv., exponentiual stretch)) *exist if and only if* $\mathcal{BPP} = \mathcal{P}$. Furthermore, such pseudorandom generators may use a seed of constant length (i.e., a two-bit long random seed).

Applying the same approach to hitting set generators, we derive a notion of a *targeted hitter*, which is adequate for derandomizing $\mathcal{RP}$. Specifically, a targeted hitter is a deterministic polynomial-time algorithm that, on input a circuit that accepts most strings of a certain length, finds a string that satisfies this circuit. Clearly, such a targeted hitter implies that $\mathcal{RP} = \mathcal{P}$, which in turn implies $\mathcal{BPP} = \mathcal{P}$ (see, e.g., [3, §6.1.3.2]). Thus, *targeted hitters exist if and only if targeted canonical derandomizers exist.*

[1]More accurately, for any $S \in \mathcal{BPP}$ and every polynomial p, there exists a deterministic polynomial-time A such that no probabilistic p-time algorithm F can find (with probability exceeding $1/p$) an input on which A errs; that is, the probability that $F(1^n)$ equals an n-bit string x such that $A(x) \neq \chi_S(x)$ is at most $1/p(n)$, where χ_S is the characteristic function of S.

Organization. For sake of self-containment, we recall (in Sections 2 and 3) the preliminaries and background that forms the basis for the current work. These parts are reproduced from [4], and contain the prior notion of targeted canonical derandomizers (see Definition 3.2). The new part of this work start in Section 3.3, which presents the new notion of targeted canonical derandomizers (see Definition 3.3). Sections 4 and 5 present the (modest) technical contribution of this work, and Section 6 de-constructs them.

2 Preliminaries

In Section 2.1, we review the notion of promise problems, while adapting it to the context of search problems, and in Section 2.2 we define "BPP search problem" (while warning that the definition is not straightforward). The entire section is reproduced from [4].

Standard notation. For a natural number n, we let $[n] \overset{\text{def}}{=} \{1, 2, ..., n\}$ and denote by U_n a random variable that is uniformly distributed over $\{0,1\}^n$. When referring to the probability that a uniformly distributed n-bit long string hits a set S, we shall use notation such as $\Pr[U_n \in S]$ or $\Pr_{r\in\{0,1\}^n}[r \in S]$.

2.1 Promise problems

We rely heavily on the formulation of promise problems (introduced in [2]). We believe that, in general, the formulation of promise problems is far more suitable for any discussion of feasibility results. The original formulation of [2] refers to decisional problems, but we shall also extend it to search problem.

In the setting of decisional problems, a promise problem, denoted $\langle P, Q\rangle$, consists of a promise (set), denoted P, and a question (set), denoted Q, such that the problem $\langle P, Q\rangle$ is defined as *given an instance $x \in P$, determine whether or not $x \in Q$.* That is, the solver is required to distinguish inputs in $P \cap Q$ from inputs in $P \setminus Q$, and nothing is required in case the input is outside P. Indeed, an equivalent formulation refers to two disjoint sets, denoted Π_{YES} and Π_{NO}, of YES- and NO-instances, respectively. We shall actually prefer to present promise problems in these terms; that is, as pairs $(\Pi_{\text{YES}}, \Pi_{\text{NO}})$ of disjoint sets. Indeed, standard decision problems appear as special cases in which $\Pi_{\text{YES}} \cup \Pi_{\text{NO}} = \{0,1\}^*$. In the general case, inputs outside of $\Pi_{\text{YES}} \cup \Pi_{\text{NO}}$ are said to violate the promise.

Unless explicitly stated otherwise, all decisional problems discussed in this work are actually promise problems, and $\mathcal{P}, \mathcal{BPP}$ etc denote the corresponding classes of promise problems. For example, $(\Pi_{\text{YES}}, \Pi_{\text{NO}}) \in \mathcal{BPP}$ if *there exists a probabilistic polynomial-time algorithm A such that for every $x \in \Pi_{\text{YES}}$ it holds that* $\Pr[A(x)=1] \geq 2/3$*, and for every $x \in \Pi_{\text{NO}}$ it holds that* $\Pr[A(x)=0] \geq 2/3$.

2.2 BPP search problem

Typically, search problems are captured by binary relations that determine the set of valid instance-solution pairs. For a binary relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$, we denote by $R(x) \stackrel{\text{def}}{=} \{y : (x,y) \in R\}$ the set of valid solutions for the instance x, and by $S_R \stackrel{\text{def}}{=} \{x : R(x) \neq \emptyset\}$ the set of instances having valid solutions. Solving a search problem R means that given any $x \in S_R$, we should find an element of $R(x)$ (whereas, possibly, we should indicate that no solution exists if $x \notin S_R$).

The definition of "BPP search problems" is supposed to capture search problems that can be solved efficiently, when random steps are allowed. Intuitively, we do not expect randomization to make up for more than an exponential blow-up, and so the naive formulation that merely asserts that solutions can be found in probabilistic polynomial-time is not good enough. Consider, for example, the relation R such that $(x,y) \in R$ if $|y| = |x|$ and for every $i < |x|$ it holds that $M_i(x) \neq y$, where M_i is the i^{th} deterministic machine (in some fixed enumeration of such machines). Then, the search problem R can be solved by a probabilistic polynomial-time algorithm (which, on input x, outputs a uniformly distributed $|x|$-bit long string), but cannot be solved by any deterministic algorithm (regardless of its running time).

What is missing in the naive formulation is any reference to the "complexity" of the solutions found by the solver, let alone to the complexity of the set of all valid solutions. We just postulate the latter (i.e., that the set of all valid instance-solutions pairs is easily recognizable). Actually[2], we generalize the treatment to search problems with a promise, where the promise allows to possibly discard some instance-solution pairs. (At first reading, the reader may assume that $R_{\text{NO}} = \{0,1\}^* \setminus R_{\text{YES}}$.)

Definition 2.1 (BPP search problems): *Let R_{YES} and R_{NO} be two disjoint binary relations. We say that $(R_{\text{YES}}, R_{\text{NO}})$ is a* BPP-search problem *if the following two conditions hold.*

[2] See motivational discussion in [4, Sec. 3.1].

1. *The decisional problem represented by* $(R_{\rm YES}, R_{\rm NO})$ *is solvable in probabilistic polynomial-time; that is, there exists a probabilistic polynomial-time algorithm* V *such that for every* $(x, y) \in R_{\rm YES}$ *it holds that* $\Pr[V(x, y)=1] \geq 2/3$*, and for every* $(x, y) \in R_{\rm NO}$ *it holds that* $\Pr[V(x, y)=1] \leq 1/3$*.*

2. *There exists a probabilistic polynomial-time algorithm* A *such that, for every* $x \in S_{R_{\rm YES}}$*, it holds that* $\Pr[A(x) \in R_{\rm YES}(x)] \geq 2/3$*, where* $R_{\rm YES}(x) = \{y : (x, y) \in R_{\rm YES}\}$ *and* $S_{R_{\rm YES}} = \{x : R_{\rm YES}(x) \neq \emptyset\}$*.*

We may assume, without loss of generality, that, for every x such that $R_{\rm NO}(x) = \{0,1\}^*$, it holds that $\Pr[A(x) = \perp] \geq 2/3$, since algorithm A can avoid outputting invalid solutions (i.e., elements of $R_{\rm NO}(x)$) by checking them using algorithm V. Note that the algorithms postulated in Definition 2.1 allow for finding valid solutions (i.e., elements of $R_{\rm YES}(x)$) as well as distinguishing valid solutions (i.e., elements of $R_{\rm YES}(x)$) from invalid ones (i.e., elements of $R_{\rm NO}(x)$), but they do not offer a way of increasing the probability of finding valid solutions (since they do not provide a way of distinguishing elements of $R_{\rm YES}(x)$ from elements of $\{0,1\}^* \setminus (R_{\rm YES}(x) \cup R_{\rm NO}(x))$.

3 Definitional Treatment

For sake of clarity and perspective, we start by reviewing the standard definition of (non-uniformly strong) canonical derandomizer (cf., e.g., [3, Sec. 8.3.1]). Next, we review the notion of a targeted canonical derandomizer that was introduced in [4, Sec. 4.4], and finally we present the new definition of a targeted canonical derandomizer. The first two subsections are reproduced from [4].

3.1 The standard (non-uniformly strong) definition

Recall that in order to "derandomize" a probabilistic polynomial-time algorithm A, we first obtain a functionally equivalent algorithm A_G that uses a pseudorandom generator G in order to reduce the randomness-complexity of A, and then take the majority vote on all possible executions of A_G (on the given input). That is, we scan all possible outcomes of the coin tosses of $A_G(x)$, which means that the deterministic algorithm will run in time that is exponential in the randomness complexity of A_G. Thus, it suffices to have a pseudorandom generator that can be evaluated in time that is exponential in its seed length (and polynomial in its output length).

In the standard setting, algorithm A_G has to maintain A's input-output behavior on all (but finitely many) inputs, and so the pseudorandomness property of G should hold with respect to distinguishers that receive non-uniform advice (which models a potentially exceptional input x on which $A(x)$ and $A_G(x)$ are sufficiently different). Without loss of generality, we may assume that A's running-time is linearly related to its randomness complexity, and so the relevant distinguishers may be confined to linear time. Similarly, for simplicity (and by possibly padding the input x), we may assume that both complexities are linear in the input length, $|x|$. (Actually, for simplicity we shall assume that both complexities just equal $|x|$, although some constant slackness seems essential.) Finally, since we are going to scan all possible random-pads of A_G and rule by majority (and since A's error probability is at most $1/3$), it suffices to require that for every x it holds that $|\Pr[A(x) = 1] - \Pr[A_G(x) = 1]| < 1/6$. This leads to the pseudorandomness requirement stated in the following definition.

Definition 3.1 (canonical derandomizers, standard version [3, Def, 8.14])[3]: *Let $\ell : \mathsf{N} \to \mathsf{N}$ be a function such that $\ell(n) > n$ for all n. A* canonical derandomizer of stretch *ℓ is a* deterministic *algorithm G that satisfies the following two conditions.*

(generation time): *On input a k-bit long seed, G makes at most* $\text{poly}(2^k \cdot \ell(k))$ *steps and outputs a string of length $\ell(k)$.*

(pseudorandomness): *For every* (deterministic) *linear-time algorithm D, all sufficiently large k and all $x \in \{0,1\}^{\ell(k)}$, it holds that*

$$|\Pr[D(x, G(U_k)) = 1] - \Pr[D(x, U_{\ell(k)}) = 1]| < \frac{1}{6} \qquad (1)$$

The algorithm D represents a potential distinguisher, which is given two $\ell(k)$-bit long strings as input, where the first string (i.e., x) represents a (non-uniform) auxiliary input and the second string is sampled either from $G(U_k)$ or from $U_{\ell(k)}$. When seeking to derandomize a linear-time algorithm A, the first string (i.e., x) represents a potential main input for A, whereas the second string represents a possible sequence of coin tosses of A (when invoked on a generic (primary) input x of length $\ell(k)$).

3.2 The original notion of targeted generators

Our main focus in [4] was on the standard notion of a uniform canonical derandomizer (as introduced in [5]), which was shown to exist (with exponential stretch) exist if and only if $\mathcal{BPP}$ is *effectively* in $\mathcal{P}$ (in the sense

[3]To streamline our exposition, we preferred to avoid the standard additional step of replacing $D(x, \cdot)$ by an arbitrary (non-uniform) Boolean circuit of quadratic size.

that it is *infeasible* to find an input on which the polynomial-time derandomized algorithm errs). Still, seeking a notion of a canonical derandomizer that can be shown to exist if and onbly if $\mathcal{BPP} = \mathcal{P}$ (*proper*), we suggested the following notion of a targeted canonical derandomizer, where both the generator and the distinguisher are presented with the same auxiliary input (or "target").

Definition 3.2 (targeted canonical derandomizers, [4, Def. 4.10]): *Let $\ell : \mathsf{N} \to \mathsf{N}$ be a function such that $\ell(n) > n$ for all n. A* targeted canonical derandomizer of stretch *ℓ is a deterministic algorithm G that satisfies the following two conditions.*

(generation time): *On input a k-bit long seed and an $\ell(k)$-bit long auxiliary input, G makes at most* $\text{poly}(2^k \cdot \ell(k))$ *steps and outputs a string of length $\ell(k)$.*

(pseudorandomness (targeted)): *For every* (deterministic) *linear-time algorithm D, all sufficiently large k and all $x \in \{0,1\}^{\ell(k)}$, it holds that*

$$|\Pr[D(x, G(U_k, x)) = 1] - \Pr[D(x, U_{\ell(k)}) = 1]| < \frac{1}{6} \quad (2)$$

Definition 3.1 is obtained from Definition 3.2 by mandating that G ignores s (i.e., $G(s,x) = G'(s)$). On the other hand, Definition 3.2 is a special case of related definitions that appeared in [8, Sec. 2.4]. Specifically, Vadhan [8] studied auxiliary-input pseudorandom generators (of the general-purpose type [1, 9]), while offering a general treatment in which pseudorandomness needs to hold for an arbitrary set of targets (i.e., $x \in I$ for some set $I \subseteq \{0,1\}^*$, whereas in Definition 3.2 we mandate $I = \{0,1\}^*$).[4]

The notion of a targeted canonical derandomizer is not as odd as it looks at first glance. Indeed, the generator is far from being general-purpose (i.e., it is tailored to a specific x), but this merely takes to (almost) the limit the insight of Nisan and Wigderson regarding relaxations that are still useful towards derandomization [6]. Indeed, even if we were to fix the distinguisher D, constructing a generator that just fools $D(x, \cdot)$ is not straightforward, because we need to find a suitable "fooling set" deterministically (in polynomial-time). The latter sentence (which is also reproduced from [4]), leads to the new definition.

[4]His treatment vastly extends the original notion of auxiliary-input one-way functions put forward in [7].

3.3 The new notion of targeted generators

Indeed, we suggest to consider canonical derandomizers that fool a single distinguisher, which is presented to them as input. The distinguisher is presented as a (deterministic) circuit, which determines the length of the sequence that the generator ought to produce. Thus, we no longer use a stretch function in our definitions. Instead, the seed length may be a function of the length of the output sequence, but it turns out that we may just use a fixed seed length (for all possible output lengths). We thus simplify our exposition by just using a fixed seed length.[5]

Definition 3.3 (targeted canonical derandomizers, revised): *A* targeted canonical derandomizer (with seed length k) *is a deterministic algorithm G that satisfies the following two conditions.*

(generation time): *On input a k-bit long seed and a circuit C with ℓ input bits, algorithm G makes at most* $\text{poly}(|\langle C\rangle|)$ *steps and outputs a string of length ℓ, where $\langle C\rangle$ denotes the description of the circuit C.*

(pseudorandomness (targeted)): *The* (ℓ-bit input) *circuit C cannot distinguish $G(U_k, \langle C\rangle)$ from U_ℓ; that is,*

$$|\Pr[C(G(U_k, \langle C\rangle)) = 1] - \Pr[C(U_\ell) = 1]| < \frac{1}{6} \tag{3}$$

A version of Definition 3.3 (in which k is logarithmic in the size of the circuit and $\ell(k) = \exp(k)$)[6] can be obtained as a special case of Definition 3.2 by replacing the generic (linear-time) D with a specific (linear-time) algorithm; the circuit evaluation algorithm E (i.e., $E(\langle C\rangle, y) = C(y)$).[7]

Indeed, Definition 3.3 takes the approach of [6] to its logical conclusion: The derandomization of algorithm A with respect to input x just yields a single circuit $C_x(\cdot) = A(x, \cdot)$ that we need to fool, and Definition 3.3 (even more than Definition 3.2) is tailored to just do that. Indeed, the existence of a generator (as in Definition 3.3), even with a seed length that is logarithmic in the circuit size, implies $\mathcal{BPP} = \mathcal{P}$ (see proof of Theorem 4.1).

[5] Indeed, in general, one may allow k to be a function of the size of the circuit.

[6] In general, using a non-decreasing function $k : \mathsf{N} \to \mathsf{N}$, we need to couple it with a function $\ell : \mathsf{N} \to \mathsf{N}$ such that $s \le \ell(k(s)) \le \text{poly}(s)$ for every $s \in \mathsf{N}$, because these pair of functions allow to handle circuits of sizes in $\{\ell(k(s)) : s \in \mathsf{N}\}$. So, actually, we use $k(s) = \lceil \log_b s\rceil$ and $\ell(k) = b^k$, for any constant $b > 1$, which implies $\ell(k(s)) < b^{1+\log_b s} = O(s)$. Hence, we may even have $k(s) = \lceil \log_b \log_b n\rceil$ (since $b^{b^{k(s)}} = b^{b \log_b s} = s^b$), but not significantly smaller.

[7] This requires using a slightly redundent description of circuits such that evaluating them can be done in linear-time.

4 The Main Result

Building on the ideas of [4], we prove the following

Theorem 4.1 (main equivalence): *Targeted canonical derandomizers* (as per Definition 3.3) *exist if and only if $\mathcal{BPP} = \mathcal{P}$. Furthermore, seed length two suffices.*

It follows that targeted canonical derandomizers as per Definition 3.3 exist if and only if generators as in Definition 3.2 (with exponential stretch) exist, since the latter also exist if and only if $\mathcal{BPP} = \mathcal{P}$ (see [4, Thm. 4.11]).

Proof: Using any targeted canonical derandomizer we obtain $\mathcal{BPP} = \mathcal{P}$ (by just feeding to the targeted canonical derandomizer the circuit that results from combining the randomized algorithm with the relevant input). That is, let A be a probabilistic polynomial-time algorithm for deciding a promise problem $\Pi = (\Pi_{\rm YES}, \Pi_{\rm NO})$, and let G be a targeted canonical derandomizer with seed length $k = O(1)$. We first consider the probabilistic polynomial-time A' that, on input x, constructs the circuit $C_x(\cdot) = A(x, \cdot)$, and outputs $A(x, G(U_k, \langle C_x \rangle))$. Clearly, if $x \in \Pi_{\rm YES}$, then $\Pr[A'(x, U_k) = 1] > 2/3 - 1/6 = 1/2$ (resp., if $x \in \Pi_{\rm NO}$, then $\Pr[A'(x, U_k) = 1] < 1/2$). Next, scanning all possible k-bit long random inputs to $A'(x)$ and ruling by majority, we obtain the desired deterministic algorithm, and $\Pi \in \mathcal{P}$ follows.

Turning to the opposite direction, we construct a targeted canonical derandomizer (with constant seed length) based on $\mathcal{BPP} = \mathcal{P}$. We do so by following the approach of [4]; that is, we first show that constructing a targeted canonical derandomizer is a BPP-search problem, which is reducible to a decisional BPP problem (by [4, Thm. 3.5]), which yields a deterministic construction (since $\mathcal{BPP} = \mathcal{P}$ by the hypothesis). Details follow.

The BPP-search problem. We first detail a BPP-search problem, denoted $(R_{\rm YES}, R_{\rm NO})$, that captures the desired construction (for seed length $k = 2$). This promise problem refers to pairs of the form $(\langle C \rangle, \overline{s})$ such that, for some ℓ, the string $\langle C \rangle$ describes a circuit with ℓ input bits and $\overline{s} = (s_1, ..., s_4)$ is a quadruple of ℓ-bit long strings. We place a pair $(\langle C \rangle, \overline{s})$ in $R_{\rm YES}$ if the difference between $\Pr_{i \in [4]}[C(s_i) = 1]$ and $\Pr[C(U_\ell) = 1]|$ is smaller than 0.16 and place it in $R_{\rm NO}$ if the difference is at least $1/6 > 0.16$. That is:

- $(\langle C \rangle, \overline{s}) \in R_{\rm YES}$ if and only if $|\Pr_{i \in [4]}[C(s_i) = 1] - \Pr[C(U_\ell) = 1]| < 0.16$.
- $(\langle C \rangle, \overline{s}) \in R_{\rm NO}$ if and only if $|\Pr_{i \in [4]}[C(s_i) = 1] - \Pr[C(U_\ell) = 1]| \geq 1/6$.

Indeed, we intentionally left a gap between the pairs in the two cases. Hence, a probabilistic polynomial-time can distinguish elements of R_{YES} from elements of R_{NO}, and so showing that $(R_{\text{YES}}, R_{\text{NO}})$ is a BPP-search problem amounts to detailing a suitable probabilistic polynomial-time algorithm for finding valid solutions. Such an algorithm is given a circuit C with ℓ input bits, and needs to find $\overline{s}$ such that $(\langle C\rangle, \overline{s}) \in R_{\text{YES}}$. This can be done as follows:

1. Using a constant number of samples, approximate $p_C \stackrel{\text{def}}{=} \Pr[C(U_\ell)=1]$ such that the approximation $\tilde{p}_C$ satisfies $\Pr[|\tilde{p}_C - p_C| > 0.01] < 0.01$.

2. Let $i_C = \lfloor 4 \cdot \tilde{p}_C \rceil \in \{0,1,2,3,4\}$, where $\lfloor \alpha \rceil$ denotes the integer closest to $\alpha \in \mathsf{R}$. Note that $\frac{i_C}{4}$ is in the interval $[\tilde{p}_C - \frac{1}{8}, \tilde{p}_C + \frac{1}{8}]$.

   If $i_C \in \{1,2,3\}$, then (using a constant number of samples) find strings $x_0, x_1 \in \{0,1\}^\ell$ such that $C(x_\sigma) = \sigma$ for every $\sigma \in \{0,1\}$. If $i_C = 0$ (resp., $i_C = 4$), then just find a string x_0 (resp., x_1) as above.

3. For $i = 1, ..., 4$, let $s_i = x_1$ if $i \leq i_C$ and let $s_i = x_0$ otherwise (i.e., if $i > i_C$). Recall that $\overline{s} = (s_1, s_2, s_3, s_4)$.

   (Indeed, $\overline{s}$ contains i_C occurances of x_1 and $4 - i_C$ occurances of x_0.)

Note that the probability that either of the first two steps fails can be upper bounded by 0.02. Otherwise, we have $|\tilde{p}_C - p_C| > 0.01$ and $\Pr_{i\in[4]}[C(s_i) = 1] = i_C/4$. Recalling that $|i_C - \tilde{p}_C| \leq 1/8$, it follows that (*in this case*) it holds that $|\Pr_{i\in[4]}[C(s_i)=1] - p_C| \leq 0.125 + 0.01 = 0.135$. Hence (overall), we have $|\Pr_{i\in[4]}[C(s_i)=1] - p_C| \leq 0.135 + 0.02 < 0.16$, as desired.

Next we reduce the foregoing BPP-search problem to a BPP decisional problem, by just invoking the following result of [4].

Reducing search to decision – [4, Thm. 3.5]: *For every BPP-search problem* $(R_{\text{YES}}, R_{\text{NO}})$*, there exists a binary relation* R *such that* $R_{\text{YES}} \subseteq R \subseteq (\{0,1\}^* \times \{0,1\}^*) \setminus R_{\text{NO}}$ *and solving the search problem of* R *is deterministically reducible to some decisional problem in* $\mathcal{BPP}$*, denoted* Π.

Applying [4, Thm. 3.5] to a BPP-search problem $(R_{\text{YES}}, R_{\text{NO}})$, we obtain a deterministic reduction of the construction of the desired pseudorandom generator to some promise problem in $\mathcal{BPP}$; indeed, the key observation is that whenever C has a solution (i.e., there exists $\overline{s}$ such that $(\langle C\rangle, \overline{s}) \in R_{\text{YES}}$) the reduction yields a sequence $\overline{s} = (s_1, ..., s_4)$ such that $(\langle C\rangle, \overline{s}) \notin R_{\text{NO}}$, which implies that $|\Pr_{i\in[4]}[C(s_i)=1] - p_C| < 1/6$.

Next, using the hypothesis $\mathcal{BPP} = \mathcal{P}$, we obtain a deterministic polynomial-time algorithm for finding such a sequence $\overline{s} = (s_1, ..., s_4)$, and the generator is defined by letting $G(i) = s_i$ (for every $i \in [4] \equiv \{0,1\}^2$). The theorem follows. $\blacksquare$

Remark 4.2 (on the distribution produced by the foregoing targeted canonical generator): *We observe that the targeted canonical generator constructed in the proof of Theorem 4.1 produces a distribution with at most two elements in its support, which can be shown to be the very minimum support size for any targeted canonical generator. Furthermore, this generator uses a seed of length $k = 2$, which is an artifact of $k = \lceil \log_2(3) \rceil$, where $1/3$ is twice the desired distinguishing gap. In general, when dealing with a distinguishing gap of $\delta < 1/2$, it suffices to use a seed of length $k = \lceil \log_2(1/2\delta) \rceil$, and such a generator suffices for derandomizing algorithms of error probability $0.5 - \delta$. Thus, we may use $\delta = 0.3$, and obtain a generator that uses a single bit seed* (and suffices for derandomizing algorithms of error probability 0.2).

5 Targeted Hitters

In this section we merely detail the last paragraph of the introduction. We first adapt Definition 3.3 to the notion of hitting set generators, while observing that when targeting a single circuit there is no need to output a set of possible strings (since we may test these strings and just output one string that satisfies the circuit).

Definition 5.1 (targeted hitters): *A* targeted hitter *is a deterministic algorithm H that satisfies the following two conditions.*

(generation time): *On input a circuit C with ℓ input bits, algorithm H makes at most $\text{poly}(|\langle C \rangle|)$ steps and outputs a string of length ℓ.*

(hitting (targeted)): *If $\Pr[C(U_\ell) = 1] > 1/2$, then $x \leftarrow H(\langle C \rangle)$ satisfies C* (i.e., $C(x) = 1$).

Indeed, any targeted canonical derandomizer (as per Definition 3.3) yields a targeted hitter. While the converse is less clear, it can be shown to hold by combining Theorem 4.1 with the fact that any targeted hitter implies $\mathcal{RP} = \mathcal{P}$, which in turn implies $\mathcal{BPP} = \mathcal{P}$ (e.g., since $\mathcal{BPP} = \mathcal{RP}^{\mathcal{RP}}$, see, e.g., [3, §6.1.3.2]). Thus we get:

Theorem 5.2 (summary): *The following four conditions are equivalent:*

1. *There exist targeted canonical derandomizers* (as per Definition 3.3).
2. *There exist targeted hitters.*

3. $\mathcal{BPP} = \mathcal{P}$.

4. $\mathcal{RP} = \mathcal{P}$.

Indeed, the proof outlined above takes the route (2) $\Rightarrow$ (4) $\Rightarrow$ (3) $\Rightarrow$ (1) $\Rightarrow$ (2). We comment that we do not see a direct proof of (4) $\Rightarrow$ (2) (i.e., a proof that does not pass via (4) $\Rightarrow$ (3)).

6 Reflections (or de-construction)

The new definition of a targeted canonical derandomizer (i.e., Definition 3.3) implicitly combines two tasks that need to be performed in deterministic polynomial-time:

1. Approximating the acceptance probability of circuits; that is, given a circuit C with ℓ input bits, the task is to (deterministically) approximate $\Pr[C(U_\ell)=1]$ up to $\pm 1/6$.

2. Finding an input that evaluates to the majority value; that is, given a circuit C with ℓ input bits, the task is to (deterministically) find an ℓ-bit string x such that $C(x) = \sigma$, where $\Pr[C(U_\ell)=\sigma] > 1/2$.

Indeed, the second task coincides with the task underlying the definition of a targeted hitter, whereas the first task is directly implied (only) by a targeted canonical derandomizer. Furthermore, deterministic polynomial-time algorithms for performing both tasks yield a targeted canonical derandomizer.[8] Interestingly, our construction of a targeted canonical derandomizer (based on $\mathcal{BPP} = \mathcal{P}$) implicitly uses a deterministic reduction of the second task to the first task, which is in turn reduced to $\mathcal{BPP}$.

The foregoing reductions are actually implicit in the proof of [4, Thm. 3.5]. Specifically, using a sufficiently good approximation of the acceptance probability of a circuit, we may find an input that satisfies the circuit by extending a prefix of such an input bit-by-bit (while making sure that the fraction of satisfying continuations is sufficiently large). Indeed, note that each of the aforementioned tasks can be used to solve a generalized version of this task that refers to an arbitrary threshold $\epsilon > 0$, provided that the running-time is allowed to depend (polynomially) on $1/\epsilon$. (In the case of the second task, the generalization is states as *given a circuit C with ℓ input bits such that* $\Pr[C(U_\ell)=1] > \epsilon$*, find an ℓ-bit string x such that $C(x) = \sigma$.*)

[8]This requires an auxiliary construction; specifically, if $\max_\sigma\{\Pr[C(U_\ell)=\sigma]\} < 5/6$, then we should also find an ℓ-bit string x' such that $C(x') = 1 - \sigma$. Assuming (w.l.o.g.) that $\sigma = 1$, this can be done by using the circuit $C'(x_1, x_2, x_3, x_4) = \bigwedge_{i\in[4]} C(x_i)$, since $(5/6)^4 < 1/2$.

Acknowledgments

The current note arised from questions posed to me during my presentation of [4] at the Institut Henri Poincare (Paris).

References

[1] M. Blum and S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SICOMP*, Vol. 13, pages 850–864, 1984. Preliminary version in *23rd FOCS*, pages 80–91, 1982.

[2] S. Even, A.L. Selman, and Y. Yacobi. The Complexity of Promise Problems with Applications to Public-Key Cryptography. *Inform. and Control*, Vol. 61, pages 159–173, 1984.

[3] O. Goldreich. *Computational Complexity: A Conceptual Perspective.* Cambridge University Press, 2008.

[4] O. Goldreich. In a World of P=BPP. *ECCC*, TR10-135, 2010. See also in *Studies in Complexity and Cryptography*, Lecture Notes in Computer Science Vol. 6650, Springer, 2011.

[5] R. Impagliazzo and A. Wigderson. Randomness vs. Time: Derandomization under a uniform assumption. *JCSS*, Vol. 63 (4), pages 672–688, 2001. Preliminary version in *39th FOCS*, 1998.

[6] N. Nisan and A. Wigderson. Hardness vs Randomness. *JCSS*, Vol. 49, No. 2, pages 149–167, 1994. Preliminary version in *29th FOCS*, 1988.

[7] R. Ostrovsky and A. Wigderson. One-Way Functions are Essential for Non-Trivial Zero-Knowledge. In *2nd Israel Symp. on Theory of Computing and Systems*, IEEE Comp. Soc. Press, pages 3–17, 1993.

[8] S. Vadhan. An Unconditional Study of Computational Zero Knowledge. *SICOMP*, Vol. 36 (4), pages 1160–1214, 2006. Preliminary version in *45th FOCS*, 2004.

[9] A.C. Yao. Theory and Application of Trapdoor Functions. In *23rd FOCS*, pages 80–91, 1982.