In a World of P=BPP

Oded Goldreich*
Department of Computer Science
Weizmann Institute of Science
Rehovot, Israel.
oded.goldreich@weizmann.ac.il

December 23, 2010

Abstract

We show that proving results such as $\mathcal{BPP} = \mathcal{P}$ essentially necessitate the construction of suitable pseudorandom generators (i.e., generators that suffice for such derandomization results). In particular, the main incarnation of this equivalence refers to the standard notion of uniform derandomization and to the corresponding pseudorandom generators (i.e., the standard uniform notion of "canonical derandomizers"). This equivalence bypasses the question of which hardness assumptions are required for establishing such derandomization results, which has received considerable attention in the last decade or so (starting with Impagliazzo and Wigderson [*JCSS*, 2001]).

We also identify a natural class of search problems that can be solved by deterministic polynomial-time reductions to $\mathcal{BPP}$. This result is instrumental to the construction of the aforementioned pseudorandom generators (based on the assumption $\mathcal{BPP} = \mathcal{P}$), which is actually a reduction of the "construction problem" to $\mathcal{BPP}$.

Caveat: Throughout the text, we abuse standard notation by letting $\mathcal{BPP}, \mathcal{P}$ etc denote classes of promise problems. We are aware of the possibility that this choice may annoy some readers, but believe that promise problem actually provide the most adequate formulation of natural decisional problems.[1]

Keywords: BPP, derandomization, pseudorandom generators, promise problems, search problems, FPTAS, randomized constructions.

*Partially supported by the Israel Science Foundation (grant No. 1041/08).

[1] Actually, the common restriction of general studies of feasibility to decision problems is merely a useful methodological simplification.

Contents

1 Introduction

We consider the question of whether results such as $\mathcal{BPP} = \mathcal{P}$ *necessitate* the construction of suitable pseudorandom generators, and conclude that the answer is essentially positive. By suitable pseudorandom generators we mean generators that, in particular, imply that $\mathcal{BPP} = \mathcal{P}$. Thus, in a sense, the *pseudorandom generators approach to the BPP-vs-P Question is complete*; that is, if the question can be resolved in the affirmative, then this answer follows from the existence of suitable pseudorandom generators.

The foregoing equivalence bypasses the question of which hardness assumptions are required for establishing such derandomization results (i.e., $\mathcal{BPP} = \mathcal{P}$), which is a question that has received considerable attention in the last decade or so (see, e.g., [17, 15, 19]). Indeed, the current work would have been obsolete if it were the case that the known answers were tight in the sense that the hardness assumptions required for derandomization would suffice for the construction of the aforementioned pseudorandom generators. See further discussion in Section 1.5.

1.1 What is meant by suitable pseudorandom generators?

The term pseudorandom generator is actually a general paradigm spanning vastly different notions that range from general-purpose pseudorandom generator (a la Blum, Micali, and Yao [2, 27]) to special-purpose generators (e.g., pairwise-independence ones [3]). The common theme is that the generators are deterministic devices that *stretch* short random seeds into longer sequences that *look random* in some sense, and that their operation is *relatively efficient*. The specific incarnations of this general paradigm differ with respect to the specific formulation of the three aforementioned terms; that is, they differ with respect to the requirements regarding (1) the amount of stretching, (2) the sense in which the output "looks random" (i.e., the "pseudorandomness" property), and (3) the complexity of the generation (or rather the stretching) process.

Recall that general-purpose pseudorandom generators operate in (some fixed) polynomial-time while producing outputs that look random to *any* polynomial-time observers. Thus, the observer is more powerful (i.e., runs for more time) than the generator itself. One key observation of Nisan and Wigderson [20] is that using such general-purpose pseudorandom generators is an over-kill when the goal is to derandomize complexity classes such as $\mathcal{BPP}$. In the latter case (i.e., for derandomizing $\mathcal{BPP}$) it suffices to have a generator that runs in exponential time (i.e., time exponential in its seed's length), since our deterministic emulation of the resulting randomized algorithm is going to incur such a factor in its running-time anyhow.[2] This leads to the notion of a *canonical derandomizer*, which fools observers of fixed complexity, while taking more time to produce such fooling sequences.

Indeed, the aforementioned "suitable pseudorandom generators" are (various (standard) forms of) canonical derandomizers. Our starting point is the non-uniform notion of canonical derandomizers used by Nisan and Wigderson [20], but since we aim at "completeness results" (as formulated above), we seek uniform-complexity versions of it. Three such versions are considered in our work, and two are shown to be sufficient and necessary for *suitable derandomizations of* $\mathcal{BPP}$.

The last assertion raises the question of *what is meant by a suitable derandomization of* $\mathcal{BPP}$. The first observation is that any reasonable notion of a canonical derandomizer is also applicable to promise problems (as defined in [4]), and so our entire discussion refers to $\mathcal{BPP}$ as a class of promise problems (rather than a class of standard decision problems).[3]

[2] Recall that the resulting (randomized) algorithm uses the generator for producing the randomness consumed by the original (randomized) algorithm, which it emulates, and that our deterministic emulation consists of invoking the resulting (randomized) algorithm on all possible random-pads.

[3] Indeed, as stated upfront, we believe that, in general, promise problem actually provide the most adequate

The second observation is that standard uniform-complexity notions of canonical derandomizers would not allow to place $\mathcal{BPP}$ in $\mathcal{P}$, because rare instances that are hard to find may not lead to a violation of the pseudorandomness guarantee. The known fix, used by Impagliazzo and Wigderson in [17], is to consider "effective derandomization" in the sense that each problem $\Pi \in \mathcal{BPP}$ is approximated by some problem $\Pi' \in \mathcal{P}$ such that it is hard to find instances in the symmetric difference of Π and Π'. Our main result refers to this notion (see Sections 4.2–4.3): Loosely speaking, it asserts that canonical derandomizers (of exponential stretch) exist if and only if $\mathcal{BPP}$ is effectively in $\mathcal{P}$. We stress that this result refers to the standard notion of uniform derandomization and to the corresponding canonical derandomizers (as in [17] and subsequent works (e.g. [24])).

We also consider a seemingly novel notion of canonical derandomizers, which is akin to notions of auxiliary-input one-way functions and pseudorandom generators considered by Vadhan [26]. Here the generator is given a target string and the distribution that it produces need only be pseudorandom with respect to efficient (uniform) observers that are given this very string as an auxiliary input. We show that such canonical derandomizers (of exponential stretch) exist if and only if $\mathcal{BPP} = \mathcal{P}$; for details, see Section 4.4.

1.2 Techniques

Our starting point is the work of Goldreich and Wigderson [10], which studied pseudorandomness with respect to (uniform) deterministic observers. In particular, they show how to construct, for every polynomial p, a generator of exponential stretch that works in time polynomial in its output and fools all deterministic p-time tests of the next-bit type (a la [2]). They observe that an analogous construction with respect to general tests (i.e., deterministic p-time distinguishers) would yield some non-trivial derandomization results (e.g., any unary set in $\mathcal{BPP}$ would be placed in $\mathcal{P}$). Thus, they concluded that there is a fundamental gap between probabilistic and deterministic polynomial-time observers.[4]

Our key observation is that the gap between probabilistic observers and deterministic ones essentially disappears if $\mathcal{BPP} = \mathcal{P}$. Actually, the gap disappeared with respect to certain ways of constructing pseudorandom generators, and the construction of [10] can be shown to fall into this category. We actually prefer a more direct approach, which is more transparent and amenable to variations. Specifically, we consider a straightforward probabilistic polynomial-time construction of a pseudorandom generator; that is, we observe that a random function (with exponential stretch) enjoys the desired pseudorandomness property, but of course the problem is that it cannot be constructed deterministically.

At this point, we define a search problem that consists of finding a suitable function (or rather its image), and observe that this problem is solvable in probabilistic polynomial-time. Using the fact that the suitability of candidate functions can be checked in probabilistic polynomial-time, we are able to *deterministically reduce* (in polynomial-time) this search problem to a (decisional) problem in $\mathcal{BPP}$. Finally, using the hypothesis (i.e., $\mathcal{BPP} = \mathcal{P}$), we obtain the desired (deterministic) construction.

---

formulation of natural decisional problems (cf. [9, Sec. 2.4.1]). Furthermore, promise problems were considered in the study of derandomization when converse results were in focus (cf. [15]). An added benefit of the use of classes of promise problems is that $\mathcal{BPP} = \mathcal{P}$ implies $\mathcal{MA} = \mathcal{NP}$.

[4] In particular, they concluded that Yao's result (by which fooling next-bit tests implies pseudorandomness) may not hold in the (uniform) deterministic setting (or, actually, may be hard to establish in that context). Indeed, recall that the next-bit tests derived (in Yao's argument) from general tests (i.e., distinguishers) are probabilistic.

1.3 Additional results

The foregoing description alluded to the possibility that $\mathcal{BPP} = \mathcal{P}$ (which refers to promise problems of decisional nature) extends to search problems; that is, that $\mathcal{BPP} = \mathcal{P}$ implies that a certain class of probabilistic polynomial-time solvable search problems can be emulated deterministically. This fact, which is used in our construction of canonical derandomizers, is proven as part of our study of "BPP-search problems" (and their relation to decisional BPP problems), which seems of independent interest and importance. Other corollaries include the conditional (on $\mathcal{BPP} = \mathcal{P}$) transformation of any probabilistic FPTAS into a deterministic one, and ditto for any probabilistic polynomial-time method of contructing and verifying objects of a predetermined property. (For details see Section 3.)

Also begging are extensions of our study to general "stretch vs derandomization time" trade-off (akin to the general "hardness vs randomness" trade-off) and to the derandomization of classes such as $\mathcal{AM}$. The first extension goes through easily (see Section 5), whereas we were not able to pull off the second (see Section 6).

1.4 Reflection

Recalling that canonical derandomizers run for more time than the distinguishers that they are intended to fool, it is tempting to say that the existence of such derandomizers may follow by diagonalization-type arguments. Specifically, for every polynomial p, it should be possible to construct in (larger) polynomial time, a set of (poly(n) many) strings $S_n \subset \{0,1\}^n$ such that a string selected uniformly in S_n is $p(n)$-time indistinguishable from a totally random n-bit string.

The problem with the foregoing prophecy is that it is not clear how to carry out such a diagonalization. However, it was observed in a couple of related works (i.e., [17, 10]) that a *random choice will do*. The problem, of course, is that we need our construction to be deterministic; that is, a deterministic construction should be able to achieve this "random looking" fooling effect. Furthermore, it is not a priori clear that the hypothesis $\mathcal{BPP} = \mathcal{P}$ may help us here, since $\mathcal{BPP} = \mathcal{P}$ refers to decisional problems.[5] Indeed, it seems that the interesting question of *determining the class of problems (e.g., search problems) that can be solved by deterministic polynomial-time reductions to* $\mathcal{BPP}$ was not addressed before. Still, as stated above, we show that the aforemention "construction problem" belongs to this class, and thus the hypothesis $\mathcal{BPP} = \mathcal{P}$ allows us to derandomize the foregoing arguement.

In any case, the point is that $\mathcal{BPP} = \mathcal{P}$ enables the construction of the aforementioned type of (suitable) pseudorandom generators; that is, the very pseudorandom generators that imply $\mathcal{BPP} = \mathcal{P}$. Thus, our main result asserts that these pseudorandom generators exist if and only if $\mathcal{BPP} = \mathcal{P}$, which in our opinion is not a priori obvious. Furthermore, our proof uncovers a very tight connection between the construction of such pseudorandom generators and $\mathcal{BPP} = \mathcal{P}$. In particular, $\mathcal{BPP} = \mathcal{P}$ yields a very simple construction of such pseudorandom generators, which in turn can be seen as fulfillining the foregoing (diagonalization) prophecy.

[5] For example, obviously, even if $\mathcal{BPP} = \mathcal{P}$, there exist no deterministic algorithms for uniformly selecting a random solution to a search problem (or just tossing a coin). Interestingly, while problems of uniform generation cannot be solved deterministically, the corresponding problems of approximating the number of solutions can be solved deterministically (sometimes in polynomial-time, especially when assuming $\mathcal{BPP} = \mathcal{P}$). This seems to contradict the celebrated equivalence between these two types of problems [18] (cf. [9, §6.4.2.1]), except that the relevant direction of this equivalence is established via probabilistic polynomial-time reductions (which are inherently non-derandomizable). Going beyond the strict boundaries of complexity, we note that $\mathcal{BPP} = \mathcal{P}$ would not eliminate the essential role of randomness in cryptography (e.g., in the context of zero-knowledge (cf. [8, Sec. 4.5.1]) and secure encryption (cf. [11])).

1.5 Related work

This work takes for granted the "hardness versus randomness" paradigm, pioneered by Blum and Micali [2], and its application to the derandomization of complexity classes such as $\mathcal{BPP}$, as pioneered by Yao [27] and revised by Nisan and Wigderson [20]. The latter work suggests that a suitable notion of a pseudorandom generator – indeed, the aforementioned notion of a canonical derandomizer – provides the "King's (high)way" to derandomization of $\mathcal{BPP}$. This view was further supported by subsequent work such as [16, 17, 25], and the current work seems to suggest that this King's way is essentially the only way.

As stated up-front, this work does not address the question of which hardness assumptions are required for establishing such derandomization results (i.e., $\mathcal{BPP} = \mathcal{P}$). Recall that this question has received considerable attention in the last decade or so, starting with the aforementioned work of Impagliazzo and Wigderson in [17], and culminating in the works of Impagliazzo, Kabanets, and Wigderson [15, 19]. We refer the interested reader to [23, Sec. 1.1-1.3] for an excellent (and quite updated) overview of this line of work.

Note that the foregoing discussion refers to three possible events: The first event is the existence of a good derandomization (e.g., $\mathcal{BPP} = \mathcal{P}$), the second is the existence of certain pseudorandom generators (i.e., canonical derandomizers), and the third is the existence of certain lower bound (i.e., hardness results). The main thread of past work (e.g., [2, 27, 20, 16, 17]) goes from hardness assumptions to pseudorandom generators and further to good derandomization (e.g., $\mathcal{BPP} = \mathcal{P}$). Later work such as [17, 15] partially reverse the the hardness to derandomization implication, whereas our work only refers and reverses the second leg of the main thread (i.e., showing that $\mathcal{BPP} = \mathcal{P}$ implies certain pseudorandom generators). We comment that the reversing of the first leg (i.e., showing that pseudorandom generators imply hardness) is folklore (see, e.g., [9, Exer. 8.24]). All these implications are depicted in Figure 1.

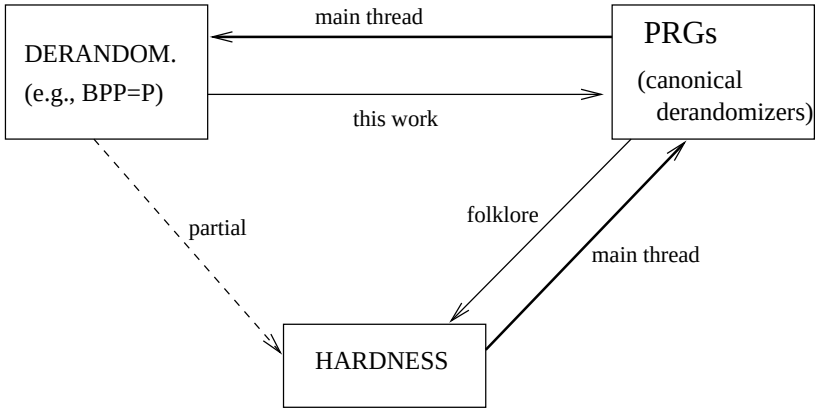

Figure 1: The three related events. The solid arrows show implications that hold for the full range of parameters, whereas the dashed arrow shows a partial implication that does not suffice for the "high end" (i.e., for pseudorandom generators that suffice for $\mathcal{BPP} = \mathcal{P}$).

Actually, both the aforementioned works [17, 15] imply results that are in the spirit of our main result, but these results refer to weak notions of derandomization, and their proofs are fundamentally different. The work of Impagliazzo and Wigderson [17] refers to the "effective infinitely often" containment of $\mathcal{BPP}$ in $\mathcal{SUBEXP}$, whereas the work of Impagliazzo, Kabanets, and Wigderson [15] refers to the (standard) containment of $\mathcal{BPP}$ in $\mathcal{NSUBEXP}/n^{\epsilon}$. In both cases, the derandomization hypotheses are shown to imply corresponding hardness results (i.e., functions in $\mathcal{EXP}$ that are not

in $\mathcal{BPP}$ or functions in $\mathcal{NEXP}$ having no polynomial size circuits, resp.), which in turn yield "correspondingly canonical" derandomizers (i.e., canonical w.r.t effectively placing $\mathcal{BPP}$ in $\mathcal{SUBEXP}$ infinitely often or placing $\mathcal{BPP}$ in $\mathcal{NSUBEXP}$, resp.).[6] Thus, in both cases, the construction of these generators (based on the relevant derandomization hypothesis) follows the "hardness versus randomness" paradigm (and, specifically, the Nisan–Wigderson framework [20]). In contrast, our constructions bypass the "hardness versus randomness" paradigm.

We also mention that the possibility of reversing the pseudorandomness-to-derandomization transformation was studied by Fortnow [5]. In terms of his work, our result indicates that in some sense Hypothesis III implies Hypothesis II.

There is a remote similarity between our search to decision reduction (see Section 3.2) and one part of the work of Aaronson *et. al.* [1]. Our reduction relies on the fact that additive error approximation of certain probabilities can be done in $\mathcal{BPP}$, and these approximations are use in our search process. Interestingly, our main application is for constructing an adequate pseudorandom set, which may be viewed as a diagonalization (w.r.t certain class of algorithms). The argument in [1, Sec. 3] relies on the fact that a multiplicative factor approximation of certain set sizes can be done in $\mathcal{AM}$, and uses these approximations to diagonalized over a certain class of circuits. (These two processes were discovered independently.)

Finally, we mention that the relation between derandomizing probabilistic search and decision classes was briefly mentioned by Reingold, Trevisan, and Vadhan in the context of $\mathcal{RL}$; see [22, Prop. 2.7].

1.6 Organization

The rather standard conventions used in this work are presented in Section 2. In Section 3 we take a close look at "BPP search problems" and their relation to $\mathcal{BPP}$. The relation between derandomizations of $\mathcal{BPP}$ and various forms of pseudorandom generators is studied in Section 4, and ramified in Section 5. A few open problems that arise naturally from this work are discussed in Section 6. The appendix presents two prior proofs of our main result, which may be of interest.

2 Preliminaries

We assume a sufficiently strong model of computation (e.g., a 2-tape Turing machine), which allows to do various simple operations very efficiently. Exact complexity classes such as $\text{DTIME}(t)$ and $\text{BPTIME}(t)$ refer to such a fixed model. We shall say that a problem Π is in $\text{DTIME}(t)$ (resp., in $\text{BPTIME}(t)$) if there exists a deterministic (resp., probabilistic) t-time algorithm that solves the problem *on all but finitely many inputs*.

We assume that all polynomials, time bounds, and stretch functions are monotonically increasing functions from N to N, which means, in particular, that they are injective. Furthermore, we assume that all these functions are time-constructible (i.e., the mapping $n \mapsto f(n)$ can be computed in less than $f(n)$ steps).

Promise problems. We rely heavily on the formulation of promise problems (introduced in [4]). We believe that, in general, the formulation of promise problems is far more suitable for any discussion of feasibility results. The original formulation of [4] refers to decision problems, but we

[6] Note that in case of [17] the generators are pseudorandom only infinitely often, whereas in the case of [15] the generators are computable in *non-deterministic* polynomial-time (with short advice). In both cases, the generators have polynomial strecth.

shall also extend it to search problem. In the original setting, 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* $\Pr[A(x)=0] \geq 2/3$.

Standard notation. For a natural number n, we let $[n] \stackrel{\text{def}}{=} \{1, 2, ..., n\}$ 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 $\Pr_{r \in \{0,1\}^n}[r \in S]$.

Negligible, noticeable, and overwhelmingly high probabilities. A function $f : \mathsf{N} \to [0,1]$ is called negligible if is decreases faster than the reciprocal of any positive polynomial (i.e., for every positive polynomial p and all sufficiently large n it holds that $f(n) < 1/p(n)$). A function $f : \mathsf{N} \to [0,1]$ is called noticeable if it is lower bound by the reciprocal of some positive polynomial (i.e., for some positive polynomial p and all sufficiently large n it holds that $f(n) > 1/p(n)$). We say that the probability of an event is overwhelmingly high if the probability of the complement event is negligible (in the relevant parameter).

3 Search Problems

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 $x \notin S_R$).

3.1 The definition

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. The first idea that comes to mind is to just postulate the latter; that is, confine ourselves to the class of search problems for which valid instance-solution pairs can be efficiently recognized (i.e., R, as a set of pairs, is in $\mathcal{BPP}$).

Definition 3.1 (BPP search problems, first attempt): *A* BPP-search problem *is a binary relation R that satisfies the following two conditions.*

1. *Membership in R is decidable in probabilistic polynomial-time.*
2. *There exists a probabilistic polynomial-time algorithm A such that, for every $x \in S_R$, it holds that* $\Pr[A(x) \in R(x)] \geq 2/3$.

We may assume, without loss of generality, that, for every $x \notin S_R$, it holds that $\Pr[A(x) = \perp] \geq 2/3$. Note that Definition 3.1 is robust in the sense that it allows for error reduction, which may not be the case if Condition 1 were to be avoided. A special case in which Condition 1 holds is when R is an NP-witness relation; in that case, the algorithm in Condition 1 is actually deterministic.

In view of our general interest in promise problems, and of the greater flexibility they offer, it makes sense to extend the treatment to promise problems. The following generalization allows a promise set not only at the level of instances, but also at the level of instance-solution pairs. Specifically, we consider disjoint sets of valid and invalid instance-solution pairs, require this promise problem to be efficiently decidable, and of course require that valid solutions be found whenever they exist.

Definition 3.2 (BPP search problems, revisited): *Let R_{YES} and R_{NO} be two disjoint binary relations. We say that $(R_{\mathrm{YES}}, R_{\mathrm{NO}})$ is a* BPP-search problem *if the following two conditions hold.*

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

We may assume, without loss of generality, that, for every x such that $(x, y) \in R_{\mathrm{NO}}$ $(\forall y)$, it holds that $\Pr[A(x) = \perp] \geq 2/3$. Note that the algorithms postulated in Definition 3.2 allow to find valid solutions as well as distinguish valid solutions from invalid ones (while guaranteeing nothing for solutions that are neither valid nor invalid).

The promise problem formulation (of Definition 3.2) captures many natural "BPP search" problems that are hard to fit into the more strict formulation of Definition 3.1. Typically, this can be done by narrowing the set of valid solutions (and possibly extending the set of invalid solutions) such that the resulting (decisional) promise problem becomes tractable. Consider for example, a search problem R (as in Definition 3.1) for which the following stronger version of Condition 2 holds.

(2') There exists a noticeable function $\mathtt{ntc}:\mathsf{N}\to[0,1]$ such that, for every $x\in S_R$ there exists $y\in R(x)$ such that $\Pr[A(x)=y]>\mathtt{ntc}(|x|)$, whereas for every $(x,y)\notin R$ it holds that $\Pr[A(x)=y]<\mathtt{ntc}(|x|)/2$.

Then, we can define $R'_{\mathrm{YES}}=\{(x,y):\Pr[A(x)=y]>\mathtt{ntc}(|x|)\}$ and $R'_{\mathrm{NO}}=\{(x,y):\Pr[A(x)=y]<\mathtt{ntc}(|x|)/2\}$, and conclude that $R'=(R'_{\mathrm{YES}},R'_{\mathrm{NO}})$ is a BPP-search problem (by using A also for Condition 1), which captures the original problem just as well. Specifically, solving the search problem R is trivially reducible to solving the search problem R', whereas we can distinguish between valid solutions to R' (which are valid for R) and invalid solutions for R (which are also invalid for R'). This is a special case of the following observation.

Observation 3.3 (companions): *Let $\Pi=(R_{\mathrm{YES}},R_{\mathrm{NO}})$ and $\Pi'=(R'_{\mathrm{YES}},R'_{\mathrm{NO}})$ be two search problems such that $S_{R'_{\mathrm{YES}}}=S_{R_{\mathrm{YES}}}$ and $R'_{\mathrm{NO}}\supseteq(\{0,1\}^*\times\{0,1\}^*)\setminus R_{\mathrm{YES}}$, which implies $R'_{\mathrm{NO}}\supseteq R_{\mathrm{NO}}$ and $R'_{\mathrm{YES}}\subseteq R_{\mathrm{YES}}$.[7] Then, solving the search problem $(R_{\mathrm{YES}},R_{\mathrm{NO}})$ is trivially reducible to solving the search problem $(R'_{\mathrm{YES}},R'_{\mathrm{NO}})$, whereas deciding membership in $(R'_{\mathrm{YES}},R_{\mathrm{NO}})$ is trivially reducible to deciding membership in $(R'_{\mathrm{YES}},R'_{\mathrm{NO}})$. We call Π' a* companion of Π*, and note that in general this notion is not symmetric.*[8]

The point of these reductions is that they allow using algorithms associated with Π' for handling Π. Specifically, we can search solutions with respect to Π' and test validity of solutions with respect to Π', while being guaranteed that nothing was lost (since we still find valid solutions for any $x\in S_{R_{\mathrm{YES}}}$, any solution in $R'_{\mathrm{YES}}(x)\subseteq R_{\mathrm{YES}}(x)$ is recognized by us as valid, and any candidate solution in $R_{\mathrm{NO}}(x)\subseteq R'_{\mathrm{NO}}(x)$ is rejected as invalid). Furthermore, by the companion condition, candidate solutions that are not valid with respect to Π are also rejected (since they are invalid w.r.t Π'); that is, if $(x,y)\notin R_{\mathrm{YES}}$ (although it needs not be in R_{NO}), then $(x,y)\in R'_{\mathrm{NO}}$ (since $R'_{\mathrm{NO}}\supseteq(\{0,1\}^*\times\{0,1\}^*)\setminus R_{\mathrm{YES}}$).

The methodology alluded to above is demonstrated next in casting any probabilistic fully polynomial-time approximation scheme (i.e., FPTAS, cf. [13]) as a search-BPP problem. A (probabilistic) FPTAS for a quantity $q:\{0,1\}^*\to\mathsf{R}^+$ is an algorithm that on input x and $\epsilon>0$ runs for $\mathrm{poly}(n/\epsilon)$ steps and, with probability at least $2/3$, outputs a value in the interval $[(1\pm\epsilon)\cdot q(x)]$. A straightforward casting of this approximation problem as a search problem refers to the binary relation Q such that $Q\stackrel{\mathrm{def}}{=}\{(\langle x,1^m\rangle,v)\in\mathsf{R}^+:|v-q(x)|\leq q(x)/m\}$. In general, however, this does not yield a BPP-search problem, since Q may not be probabilistic polynomial-time recognizable. Instead, we consider the BPP-search problem $(R_{\mathrm{YES}},R_{\mathrm{NO}})$ such that $(\langle x,1^m\rangle,v)\in R_{\mathrm{YES}}$ if $|v-q(x)|\leq q(x)/3m$ and $(\langle x,1^m\rangle,v)\in R_{\mathrm{NO}}$ if $|v-q(x)|>q(x)/m$. Indeed, on input $\langle x,1^m\rangle$ we find a solution in $R_{\mathrm{YES}}(\langle x,1^m\rangle)$ by invoking the FPTAS on input x and $\epsilon=1/3m$, and deciding the validity of a pair $(\langle x,1^m\rangle,v)$ w.r.t $(R_{\mathrm{YES}},R_{\mathrm{NO}})$ is done by obtaining a good approximation of $q(x)$ (and decid-

[7] The first conclusion (i.e., $R'_{\mathrm{NO}}\supseteq R_{\mathrm{NO}}$) follows by the fact that $R_{\mathrm{NO}}\subseteq(\{0,1\}^*\times\{0,1\}^*)\setminus R_{\mathrm{YES}}$, whereas $R'_{\mathrm{YES}}\subseteq R_{\mathrm{YES}}$ follows since $R'_{\mathrm{NO}}\subseteq(\{0,1\}^*\times\{0,1\}^*)\setminus R'_{\mathrm{YES}}$. Observation 3.3 itself relies only on these conclusions (i.e., $R'_{\mathrm{NO}}\supseteq R_{\mathrm{NO}}$ and $R'_{\mathrm{YES}}\supseteq R_{\mathrm{YES}}$) as well as on $S_{R'_{\mathrm{YES}}}\supseteq S_{R_{\mathrm{YES}}}$; the stronger condition (i.e., $R'_{\mathrm{NO}}\supseteq(\{0,1\}^*\times\{0,1\}^*)\setminus R_{\mathrm{YES}}$) is used in other applications of the notion of companion problems (see the discussion following Theorem 3.5).

[8] Actually, if $(R_{\mathrm{YES}},R_{\mathrm{NO}})$ and $(R'_{\mathrm{YES}},R'_{\mathrm{NO}})$ are companions of one another, then they are identical (since $R'_{\mathrm{NO}}=R_{\mathrm{NO}}$ and $R'_{\mathrm{YES}}=R_{\mathrm{YES}}$ must hold). Furthermore, in this case the promise is trivial, since $R'_{\mathrm{NO}}=R_{\mathrm{NO}}\supseteq(\{0,1\}^*\times\{0,1\}^*)\setminus R'_{\mathrm{YES}}$ whereas $R'_{\mathrm{NO}}\subseteq(\{0,1\}^*\times\{0,1\}^*)\setminus R'_{\mathrm{YES}}$. Also note that each problem is its own companion, and that problems with trivial promise have no other companion (i.e., if $(R'_{\mathrm{YES}},R'_{\mathrm{NO}})$ is a companion of $(R,\overline{R})$, where $\overline{R}=(\{0,1\}^*\times\{0,1\}^*)\setminus R$, then $(R'_{\mathrm{YES}},R'_{\mathrm{NO}})=(R,\overline{R})$).

ing accordingly).[9] Indeed, $(R_{\text{YES}}, R_{\text{NO}})$ is a companion of $(Q, \overline{Q})$, where $\overline{Q} = (\{0,1\}^* \times \{0,1\}^*) \setminus Q$. Thus, we obtain.

Observation 3.4 (FPTAS as BPP-search problems): *Let $q : \{0,1\}^* \to \mathsf{R}^+$ and suppose that there exists a probabilistic FPTAS for approximating q; that is, suppose that there exists a probabilistic polynomial-time algorithm A such that $\Pr[|A(x,1^m) - q(x)| \leq q(x)/m] \geq 2/3$. Then, this approximation task is trivially reducible to some search-BPP problem* (i.e., the foregoing one). *Furthermore, the probabilistic time-complexity of the latter search problem is linearly related to the probabilistic time-complexity of the original approximation problem. Moreover, this search-BPP problem is a companion of the original approximation problem.*

3.2 The reduction

One may expect that any BPP-search problem be *deterministically* reducible to some BPP decision problem. Indeed, this holds for the restricted definition of BPP-search problems as in Definition 3.1, but for the revised formulation of Definition 3.2 we only present a weaker result. Specifically, for every BPP-search problem $(R_{\text{YES}}, R_{\text{NO}})$, there exists $R \supseteq R_{\text{YES}}$ such that $R \cap R_{\text{NO}} = \emptyset$ and solving the search problem of R is deterministically reducible to some BPP decision problem.[10]

Theorem 3.5 (reducing search to decision): *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. Furthermore, the time-complexity of the reduction is linear in the probabilistic time-complexity of finding solutions for $(R_{\text{YES}}, R_{\text{NO}})$, whereas the probabilistic time-complexity of Π is the product of a quadratic polynomial and the probabilistic time-complexity of the decision procedure guaranteed for $(R_{\text{YES}}, R_{\text{NO}})$.*

Applying Theorem 3.5 to a BPP-search problem $(R_{\text{YES}}, R_{\text{NO}})$ that is a companion of some search problem $(\Psi_{\text{YES}}, \Psi_{\text{NO}})$, we obtain a deterministic reduction of solving the search problem $(\Psi_{\text{YES}}, \Psi_{\text{NO}})$ to some promise problem in $\mathcal{BPP}$, because $S_{\Psi_{\text{YES}}} = S_{R_{\text{YES}}} \subseteq S_R$ whereas $R \subseteq (\{0,1\}^* \times \{0,1\}^*) \setminus R_{\text{NO}} \subseteq \Psi_{\text{YES}}$. The argument in depicted in Figure 2.

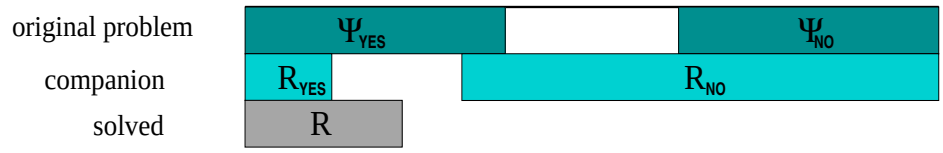

Figure 2: The reduction applied to a companion of Ψ.

Proof: Let A and V be the two probabilistic polynomial-time algorithms associated (by Definition 3.2) with the BPP-search problem $(R_{\text{YES}}, R_{\text{NO}})$, and let t_A and t_V denote their (probabilistic) time-complexities. Specifically, A is the solution-finding algorithm guaranteed by Condition 2, and V is the decision procedure guaranteed by Condition 1. Denote by $A(x,r)$ the output of algorithm

[9] That is, we invoke the FPTAS on input x and $\epsilon = 1/3m$, obtain a value $q'(x)$, which with probability at least $2/3$ is in $(1 \pm \epsilon) \cdot q(x)$, and accept if and only if $|v - q'(x)| \leq 2q(x)/3m$. Indeed, if $v \in R_{\text{YES}}(\langle x, 1^m \rangle)$, then with probability at least $2/3$ it holds that $|v - q'(x)| \leq |v - q(x)| + |q(x) - q'(x)| \leq 2q(x)/3m$, whereas if $v \in R_{\text{NO}}(\langle x, 1^m \rangle)$, then with probability at least $2/3$ it holds that $|v - q'(x)| \geq |v - q(x)| - |q(x) - q'(x)| > 2q(x)/3m$.

[10] Indeed, in the special case of Definition 3.1 (where $(R_{\text{YES}}, R_{\text{NO}})$ is a partition of the set of all pairs), it holds that $R = R_{\text{YES}}$.

A on input x and internal coins $r \in \{0,1\}^{t_A(|x|)}$, and let $V((x,y),\omega)$ denote the decision of V on input (x,y) when using coins $\omega \in \{0,1\}^{t_V(|x|+|y|)}$. Now, define

$$R \stackrel{\text{def}}{=} \left\{(x,y) : \Pr_{\omega\in\{0,1\}^{t_V(|x|+|y|)}}[V((x,y),\omega)=1] \geq 0.4\right\}, \tag{1}$$

and note that $R_{\text{YES}} \subseteq R$ and $R_{\text{NO}} \cap R = \emptyset$.

We now consider an auxiliary algorithm A'' such that $A''(x,r,\omega) \stackrel{\text{def}}{=} V((x,A(x,r)),\omega)$. Note that, for every x and r such that $(x,A(x,r)) \in R_{\text{YES}}$, it holds that $\Pr_\omega[A''(x,r,\omega)=1] \geq 2/3$, and thus, for every $x \in S_{R_{\text{YES}}}$, it holds that $\Pr_{r,\omega}[A''(x,r,\omega)=1] \geq 4/9$.

Given x, our strategy is to try to find r such that $A(x,r) \in R(x)$, by determining the bits of r one by one. We thus start with an empty prefix of r, denoted r', and in each iteration we try to extend r' by one bit. Assuming that $x \in S_{R_{\text{YES}}}$ (or rather that Eq. (2) holds for $r' = \lambda$), we try to *maintain the invariant*

$$\Pr_{r''\in\{0,1\}^{m-|r'|},\omega\in\{0,1\}^{\ell}}[A''(x,r'r'',\omega)=1] \geq \frac{4}{9} - \frac{|r'|}{25m}, \tag{2}$$

where $m = t_A(|x|)$ and $\ell = t_V(|x|+m)$. Note that if this invariant holds for $r' \in \{0,1\}^m$, then necessarily $y \stackrel{\text{def}}{=} A(x,r') \in R(x)$ (since in this case Eq. (2) implies that $\Pr_\omega[V((x,y),\omega)=1] \geq \frac{4}{9} - 0.04 > 0.4$).

Once a candidate solution $y = A(x,r')$ is found (using the corresponding $r' \in \{0,1\}^m$), we check whether y is a good solution, and output y if it is good and $\perp$ otherwise. Specifically, we test whether $(x,y) \in R$ or $(x,y) \in R_{\text{NO}}$ by making a single BPP-query (analogously to the next discussion, since for $(x,y) \in R_{\text{NO}}$ it holds that $\Pr_\omega[V((x,y),\omega)=1] \leq 1/3$).

In view of the foregoing, we focus on the design of a single iteration. Our strategy is to rely on an oracle for the promise problem $\Pi_{A''}$ that consists of YES-instances $(x,1^m,r')$ such that $\Pr_{r'',\omega}[A''(x,r'r'',\omega)=1] \geq \frac{4}{9} - \frac{|r'|-1}{25m}$ and NO-instances $(x,1^m,r')$ such that $\Pr_{r'',\omega}[A''(x,r'r'',\omega)=1] < \frac{4}{9} - \frac{|r'|}{25m}$, where in both cases the probability is taken uniformly over $r'' \in \{0,1\}^{m-|r'|}$ (and $\omega \in \{0,1\}^{\ell}$). The oracle $\Pi_{A''}$ is clearly in $\mathcal{BPP}$ (e.g., consider a probabilistic polynomial-time algorithm that on input $(x,1^m,r')$ estimates $\Pr_{r'',\omega}[A''(x,r'r'',\omega)=1]$ up to an additive term of $1/50m$ with error probability at most $1/3$, by taking a sample of $O(m^2)$ random pairs (r'',ω)).

In each iteration, which starts with some prefix r' that satisfies Eq. (2), we make a single query to the oracle $\Pi_{A''}$; specifically, we query $\Pi_{A''}$ on $(x,1^m,r'0)$. If the oracle answers positively, then we extend the current prefix r' with 0 (i.e., we set $r' \leftarrow r'0$), and otherwise we set $r' \leftarrow r'1$.

The point is that if $\Pr_{r''\in\{0,1\}^{m-|r'|},\omega}[A''(x,r'r'',\omega)=1] \geq \frac{4}{9} - \frac{|r'|}{25m}$, then there exists $\sigma \in \{0,1\}$ such that $\Pr_{r'''\in\{0,1\}^{m-|r'|-1},\omega}[A''(x,r'\sigma r''',\omega)=1] \geq \frac{4}{9} - \frac{|r'|}{25m} = \frac{4}{9} - \frac{|r'\sigma|-1}{25m}$, which means that $(x,1^m,r'\sigma)$ is a YES-instance. Thus, if Π answers negatively to the query $(x,1^m,r'0)$, then $(x,1^m,r'0)$ cannot be a YES-instance, which implies that $(x,1^m,r'1)$ is a YES-instance, and the invariance of Eq. (2) holds for the extended prefix $r'1$. On the other hand, if $\Pi = \Pi_{A''}$ answers positively to the query $(x,1^m,r'0)$, then $(x,1^m,r'0)$ cannot be a NO-instance, and the invariance of Eq. (2) holds for the extended prefix $r'0$. We conclude that each iteration of our reduction preserves the said invariance.

To verify the furthermore-part, we note that the reduction consists of $t_A(|x|)$ iterations, where in each iteration a query is made to Π and some very simple steps are taken. In particular, each query made is simply related to the previous one (i.e., can be obtained from it in constant time), and so the entire reduction has time complexity $O(t_A)$. The time complexity of Π on inputs of the form $y = (x,1^m,r')$ is $O(m^2) \cdot O(t_V(|x|+m)) = O(|y|^2 \cdot t_V(|y|))$. The theorem follows. ■

Digest. The proof of Theorem 3.5 follows the strategy of reducing NP-search problems to $\mathcal{NP}$, except that more care is required in the process. This is reflected in the invariance stated in Eq. (2) as well as in the fact that we make an essential use of promise problems (in the oracle).

3.3 Applications

As stated in the introduction, Theorem 3.5 plays a central role in establishing our main result (i.e., the reversing of the "pseudorandomness to derandomization" implication). In this section, we explore a few additional applications of Theorem 3.5. In particular, we show that $\mathcal{BPP} = \mathcal{P}$ implies a host of derandomization results that refer to computational problems that are not of the decisional type. Indeed, we shall reduce these problems to BPP-search problems and apply Theorem 3.5.

Approximations. In light of the foregoing discussion (i.e., Observation 3.4), every approximation problem that has a probabilistic FPTAS can be deterministically reduced to $\mathcal{BPP}$. Thus:

Corollary 3.6 (implication for FPTAS): *If $\mathcal{BPP} = \mathcal{P}$, then every function that has a probabilistic fully polynomial-time approximation scheme* (FPTAS) *also has such a deterministic scheme. Furthermore, for every polynomial p, there exists a polynomial p' such that if the probabilistic scheme runs in time p, then the deterministic one runs in time p'.*

The furthermore part is proved by using the furthermore parts of Observation 3.3 and Theorem 3.5 as well as a completeness feature of $\text{BPTIME}(\cdot)$. Specifically, by combining the aforementioned reductions, we infer that the approximation problem (which refers to instances of the form $\langle x, 1^m \rangle$) is (deterministically) p_1-time reducible to a problem in $\text{BPTIME}(p_2)$, where $p_1(n) = O(p(n))$ and $p_2(n) = O(n^2 \cdot p(n))$. Next, we use the fact that $\text{BPTIME}(p_2)$ has a complete problem, where completeness holds under quadratic-time reductions (which prepend the input by the original problem's description and pad it with a quadratic number of zeros).[11] The point is that this complete problem only depends on p_2, which in turn is uniquely determined by p. The hypothesis (i.e., $\mathcal{BPP} = \mathcal{P}$) implies that this $\text{BPTIME}(p_2)$-complete problem is in $\text{DTIME}(p_3)$ for some polynomial p_3, which is solely determined by p_2, and the claim follows for $p' = p_3 \circ p_1^2$. Indeed, we have also established *en passant* the following result, which is of independent interest.

Proposition 3.7 *If $\mathcal{BPP} = \mathcal{P}$, then, for every polynomial p, there exists a polynomial p' such that* $\text{BPTIME}(p) \subseteq \text{DTIME}(p')$.

Indeed, by the DTIME Hierarchy Theorem, it follows that, if $\mathcal{BPP} = \mathcal{P}$, then, *for every polynomial p, there exists a polynomial p'' such that* $\text{DTIME}(p'')$ *contains problems that are not in* $\text{BPTIME}(p)$.

Constructions of varying quality. While the foregoing discussion of approximation schemes is related to our previous proofs of the main result (see the Appendix), the following discussion is more related to the current proof (as presented in Section 4.2). We consider general construction problems, which are defined in terms of a quality function $q : \{0,1\}^* \to [0,1]$, when for a given n we need to construct an object $y \in \{0,1\}^n$ such that $q(y) = 1$. Specifically, we consider such construction problems that can be solved in probabilistic polynomial-time and have a FPTAS for evaluating the quality of candidate constructions. One interesting special case corresponds to rigid

[11] The quadratic padding of x allows $p_2(|x|)$ steps of $M(x)$ to be emulated in time $\widetilde{O}(|M| \cdot p_2(|x|))$, which is upper-bounded by $p_2((|M| + |x|)^2)$, assuming that p_2 is (say) at least quadratic.

construction problems in which the function q is Boolean (i.e., candidate constructions have either value 0 or 1). In this special case (e.g., generating an n-bit long prime) the requirement that q has a FPTAS is replaced by requiring that the set $q^{-1}(1)$ is in $\mathcal{BPP}$.

Proposition 3.8 (derandomizing some constructions): *Consider a generalized construction defined via a quality function q that has a FPTAS, and let $R_q \stackrel{\text{def}}{=} \{((1^n,1^m),y) : y \in \{0,1\}^n \wedge q(y) > 1-(1/m)\}$. Suppose that there exists a probabilistic polynomial-time algorithm that solves the search problem of R_q. Then, if $\mathcal{BPP} = \mathcal{P}$, then there exists a deterministic polynomial-time algorithm that solves the search problem of R_q.*

For example, if $\mathcal{BPP} = \mathcal{P}$, then n-bit long primes can be found in deterministic poly(n)-time. On the other hand, the treatment can be generalized to constructions that need to satisfy some auxiliary specification, captured by an auxiliary input x (e.g., on input a prime $x = P$ find a quadratic non-residue mod P). In this formulation, $R_q \stackrel{\text{def}}{=} \{((x,1^m),y) : q(x,y) > 1-(1/m)\}$, where $q\colon\{0,1\}^* \times \{0,1\}^* \to [0,1]$ can also impose length restrictions on the desired construct.

Proof: Consider the BPP-search problem $(\Pi_{\text{YES}}, \Pi_{\text{NO}})$, where $\Pi_{\text{YES}} = \{((1^n,1^m),y) : y \in \{0,1\}^n \wedge q(y) > 1-(1/2m)\}$ and $\Pi_{\text{NO}} = \{((1^n,1^m),y) : y \in \{0,1\}^n \wedge q(y) \leq 1-(1/m)\}$. Note that $(\Pi_{\text{YES}}, \Pi_{\text{NO}})$ is a companion of the search problem R_q, and apply Theorem 3.5. ■

Corollary 3.9 (a few examples): *If $\mathcal{BPP} = \mathcal{P}$, then there exist deterministic polynomial-time algorithms for solving the following construction problems.*

1. *For any fixed $c > 7/12$, on input N, find a prime in the interval $[N, N+N^c]$.*
2. *On input a prime P and 1^d, find an irreducible polynomial of degree d over $\mathrm{GF}(P)$.*
   Recall that finding a quadratic non-residue modulo P is a special case.[12]
3. *For any fixed $\epsilon > 0$ and integer $d > 2$, on input 1^n, find a d-regular n-vertex graph with second eigenvalue having absolute value at most $2\sqrt{d-1}+\epsilon$.*

The foregoing items are based on the density of the corresponding objects in a natural (easily sampleable) set. Specifically, for Item 1 we rely on the density of prime numbers in this interval [14], for Item 2 we rely on the density of irreducible polynomials [7], and for Item 3 we rely on the density of "almost Ramanujan" graphs [6].[13] In all cases there exist deterministic polynomial-time algorithms for recognizing the desired objects.

4 Canonical Derandomizers

In Section 4.1 we present and motivate the rather standard notion of a canonical derandomizer, which is the notion to which most of this work refers to. Our main result, the reversing of the pseudorandomness-to-derandomization transformation is presented in Section 4.2. One tightening, which allows to derive an equivalence, is presented in Section 4.3, which again refers to a rather standard notion (i.e., of "effectively placing $\mathcal{BPP}$ in $\mathcal{P}$"). An alternative equivalence is derived in Section 4.4, which refers to a (seemingly new) notion of a *targeted* canonical derandomizer.

[12] If the polynomial X^2+bX+c is irreducible, then so is $(X+(b/2))^2+(c-(b/2)^2)$, and it follows that $-(c-(b/2)^2)$ is a quadratic non-residue.

[13] Recall that Ramanujan graphs are known to be constructable only for specific values of d and of n.

4.1 The definition

We start by reviewing the most standard definition of canonical derandomizers (cf., e.g., [9, Sec. 8.3.1]). 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 4.1 (canonical derandomizers, standard version [9, Def, 8.14])[14]: *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* $\mathrm{poly}(2^k \cdot \ell(k))$ *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*

$$|\Pr[D(x, G(U_k)) = 1] - \Pr[D(x, U_{\ell(k)}) = 1]| < \frac{1}{6}. \tag{3}$$

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 $\ell(k)$).

Towards a uniform-complexity variant. Seeking a uniform-complexity analogue of Definition 4.1, the first thing that comes to mind is the following definition.

Definition 4.2 (canonical derandomizers, a uniform version): *As Definition 4.1, except that the original pseudorandomness condition is replaced by*

[14] 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.

(pseudorandomness, revised): *For every* (deterministic) *linear-time algorithm D, it is infeasible, given $1^{\ell(k)}$, to find a string $x \in \{0,1\}^{\ell(k)}$ such that Eq.* (3) *does not hold. That is, for every probabilistic polynomial-time algorithm F such that $|F(1^{\ell(k)})| = \ell(k)$, there exists a negligible function* $\mathtt{negl}$ *such that if $x \leftarrow F(1^{\ell(k)})$, then Eq.* (3) *holds with probability at least $1 - \mathtt{negl}(\ell(k))$.*

When seeking to derandomize a probabilistic (linear-time) algorithm A, the auxiliary algorithm F represents an attempt to find a string $x \in \{0,1\}^{\ell(k)}$ on which $A(x)$ behaves differently depending on whether it is fed with random bits (i.e., $U_{\ell(k)}$) or with pseudorandom ones produced by $G(U_k)$.

Note that if there exists a canonical derandomizer of exponential stretch (i.e., $\ell(k) = \exp(\Omega(k))$), then $\mathcal{BPP}$ is "effectively" in $\mathcal{P}$ in the sense that for every problem in $\mathcal{BPP}$ there exists a deterministic polynomial-time algorithm A such that it is infeasible to find inputs on which A errs. We hoped to prove that $\mathcal{BPP} = \mathcal{P}$ implies the existence of such derandomizers, but do not quite prove this. Instead, we prove a closely related assertion that refers to the following revised notion of a canonical derandomizer, which is implicit in [17]. In this definition, the finder F is incorporated in the distinguisher D, which in turn is an arbitrary probabilistic algorithm that is allowed some fixed polynomial-time (rather than being deterministic and linear-time).[15] (In light of the central role of this definition in the current work, we spell it out rather than use a modification on Definition 4.1 (as done in Definition 4.2).)

Definition 4.3 (canonical derandomizers, a revised uniform version): *Let $\ell, t : \mathsf{N} \to \mathsf{N}$ be functions such that $\ell(n) > n$ for all n. A t-*robust canonical derandomizer of stretch *ℓ is a* deterministic *algorithm G that satisfies the following two conditions.*

(generation time (as in Definition 4.1)): *On input a k-bit long seed, G makes at most* $\mathrm{poly}(2^k \cdot \ell(k))$ *steps and outputs a string of length $\ell(k)$.*

(pseudorandomness, revised again): *For every probabilistic t-time algorithm D and all sufficiently large k, it holds that*

$$|\Pr[D(G(U_k)) = 1] - \Pr[D(U_{\ell(k)}) = 1]| < \frac{1}{t(\ell(k))}. \tag{4}$$

Note that, on input an $\ell(k)$-bit string, the algorithm D runs for at most $t(\ell(k))$ steps.

The pseudorandomness condition implies that, for every linear-time D' and every probabilistic t-time algorithm F (such that $|F(1^n)| = n$ for every n), it holds that

$$|\Pr[D'(F(1^{\ell(k)}), G(U_k)) = 1] - \Pr[D'(F(1^{\ell(k)}), U_{\ell(k)}) = 1]| < \frac{1}{t(\ell(k))}. \tag{5}$$

Note that if, for every x, there exists a σ such that $\Pr[D'(x, U_{|x|}) = \sigma] \geq 1 - (1/3t(|x|))$ (as is the case when D' arises from an "amplified" BPP decision procedure), then the probability that

[15] Thus, Definition 4.2 and Definition 4.3 are incomparable (when the time bound t is a fixed polynomial). On the one hand, Definition 4.3 seems weaker because we effectively fix the polynomial time bound of F (which is incorporated in D). On the other hand, Definition 4.3 seems stronger because D itself is allowed to be probabilistic and run in time t (whereas in Definition 4.2 these privileges are only allowed to F, which may be viewed as a preprocessing step). Indeed, if $\mathcal{E}$ requires exponential size circuits, then there exist pseudorandom generators that satisfy one definition but not the other: On the one hand, this assumption yields the existence of a non-uniformly strong canonical pseudorandom generator (i.e., satisfying Definition 4.1) of exponential stretch [16] that is not p-robust (i.e., fails Definition 4.3), for some sufficiently large polynomial p. On the other hand, the assumption implies $\mathcal{BPP} = \mathcal{P}$, which leads to the opposite separation described at the end of Section 4.2.

$F(1^{\ell(k)})$ finds an instance $x \in \{0,1\}^{\ell(k)}$ on which $D'(x, G(U_k))$ leans in the opposite direction (i.e., $\Pr[D'(x, U_{|x|}) \neq \sigma] \geq 1/2$) is smaller than $3/t(\ell(k))$. A more general (albeit quantatively weaker) statement is proved next.

Proposition 4.4 (on the effect of canonical derandomizers): *For $t : \mathsf{N} \to \mathsf{N}$ such that $t(n) > (n \log n)^3$, let G be a t-robust canonical derandomizer of stretch ℓ. Let A be a probabilistic linear-time algorithm, and let A_G be as in the foregoing discussions* (i.e., $A_G(x,s) = A(x, G(s))$). *Then, for every probabilistic $(t/2)$-time algorithm F and all sufficiently large k, the probability that $F(1^{\ell(k)})$ hits the set $\nabla_{A,G}(k) \setminus B_A(k)$ is at most $40/t(\ell(k))^{1/3}$, where*

$$\nabla_{A,G}(k) \stackrel{\text{def}}{=} \left\{ x \in \{0,1\}^{\ell(k)} : \left| \Pr[A_G(x, U_k) = 1] - \Pr[A(x, U_{\ell(k)}) = 1] \right| > \frac{1}{3} \right\} \quad (6)$$

$$B_A(k) \stackrel{\text{def}}{=} \left\{ x \in \{0,1\}^{\ell(k)} : \frac{1}{t(\ell(k))^{1/3}} < \Pr[A(x, U_{\ell(k)}) = 1] < 1 - \frac{1}{t(\ell(k))^{1/3}} \right\}. \quad (7)$$

That is, $B_A(\cdot)$ denotes the set of inputs x on which $A(x) = A(x, U_{|x|})$ is not "almost determined" and $\nabla_{A,G}(\cdot)$ denotes the set of inputs x on which there is a significant discrepancy between the distributions $A(x)$ and $A_G(x)$.

The forgoing discussion refers to the special case in which $B_A(k) = \emptyset$. In general, if A is a decision procedure of negligible error probability (for some promise problem),[16] then A_G is essentially as good as A, since it is hard to find an instance x that matters (i.e., one on which A's error probability is negligible) on which A_G errs (with probability greater than, say, 0.4). This leads to "effectively good" derandomization of $\mathcal{BPP}$. In particular, if G has exponential stretch, then $\mathcal{BPP}$ is "effectively" in $\mathcal{P}$ (see Theorem 4.9).

Proof: Suppose towards the contradiction that there exist algorithms A and F that violate the claim. For each $\sigma \in \{0,1\}$, we consider the following probabilistic t-time distinguisher, denoted D_σ. On input r (which is drawn from either $U_{\ell(k)}$ or $G(U_k)$), the distinguisher D_σ behaves as follows.

1. Obtains $x \leftarrow F(1^{|r|})$.
2. Approximates $p(x) \stackrel{\text{def}}{=} \Pr[A(x, U_{|x|}) = \sigma]$, obtaining an estimate, denoted $\tilde{p}(x)$, such that $\Pr[|\tilde{p}(x) - p(x)| \leq t(|x|)^{-1/3}] = 1 - \mathtt{negl}(|x|)$.
3. If $\tilde{p}(x) < 1 - 2t(|x|)^{-1/3}$, then D_σ halts with output 0.
4. Otherwise (i.e., $\tilde{p}(x) \geq 1 - 2t(|x|)^{-1/3}$), D_σ invokes A on (x, r), and outputs 1 if and only if $A(x, r) = \sigma$. (Indeed, the actual input r is only used in this step.)

We stress that D_σ only approximate the value of $p(x) = \Pr[A(x, U_{|x|}) = \sigma]$ (i.e., it does not approximate the value of $\Pr[A(x, G(U_{\ell^{-1}(|x|)})) = \sigma]$, which would have required invoking G). Observe that D_σ runs for at most $t(|r|)$ steps, because the approximation of $p(x)$ amounts to $\widetilde{O}(t(|r|)^{2/3})$ invocations of $A(x)$, whereas each invocation costs $O(|r|)$ time (including the generation of truly random coins for A).

Let $q_\sigma(k)$ denote the probability that, on an $\ell(k)$-bit long input, algorithm D_σ moves to the final (input dependent) step, and note that $q_\sigma(k)$ is independent of the specific input $r \in \{0,1\}^{\ell(k)}$. Assuming that $|\tilde{p}(x) - p(x)| \leq t(|x|)^{-1/3}$ (for the string x selected at the first step), if the algorithm

[16] That is, $B_A(\cdot)$ contains only instances that violate the promise.

moves to the final step, then $p(x) > 1 - 3t(|x|)^{-1/3}$. (Similarly, if $p(x) > 1 - t(|x|)^{-1/3}$, then the algorithm moves to the final step.) Thus, the probability that $D_\sigma(U_{\ell(k)}))$ outputs 1 is at least $(1-\mathtt{negl}(\ell(k)))\cdot q_\sigma(k)\cdot(1-3t(|x|)^{-1/3})$, which is greater than $q_\sigma(k)-4t(|x|)^{-1/3}$. On the other hand, by the contradiction hypothesis, there exists a σ such that with probability at least $20t(\ell(k))^{-1/3}$, it holds that $F(1^{\ell(k)})$ hits the set $\nabla_{A,G}(k) \cap S_{\sigma,A}(k)$, where

$$S_{\sigma,A}(k) \stackrel{\text{def}}{=} \left\{ x \in \{0,1\}^{\ell(k)} : \Pr[A(x, U_{\ell(k)}) = \sigma] \geq 1 - \frac{1}{t(\ell(k))^{1/3}} \right\} \quad (8)$$

In this case (i.e., when $x \in \nabla_{A,G}(k) \cap S_{\sigma,A}(k)$) it holds that $p(x) > 1 - t(|x|)^{-1/3}$ (since $x \in S_{\sigma,A}(k)$) and $\Pr[A(x, G(U_{\ell^{-1}(|x|)})) = \sigma] < 2/3$ (since $x \in \nabla_{A,G}(k)$ and $p(x) > 1 - t(|x|)^{-1/3}$). It follows that the probability that $D_\sigma(G(U_k))$ outputs 1 is at most $(q_\sigma(k) - 20t(|x|)^{-1/3}) \cdot 1 + 20t(|x|)^{-1/3} \cdot 2/3 + \mathtt{negl}(\ell(k))$, which is smaller than $q_\sigma(k) - 5t(|x|)^{-1/3}$. Thus, we derive a contradiction to the t-robustness of G, and the claim follows. ■

4.2 The main result

Our main result is that $\mathcal{BPP} = \mathcal{P}$ implies the existence of canonical derandomizers of exponential stretch (in the sense of Definition 4.3). We conclude that seeking canonical derandomizers of exponential stretch is "complete" with respect to placing $\mathcal{BPP}$ in $\mathcal{P}$. (The same holds w.r.t "effectively" placing $\mathcal{BPP}$ in $\mathcal{P}$, see Theorem 4.9.)

Theorem 4.5 (on the completeness of canonical derandomization): *If $\mathcal{BPP} = \mathcal{P}$, then, for every polynomial p, there exists a p-robust canonical derandomizer of exponential stretch.*

The proof of Theorem 4.5 is inspired by the study of pseudorandomness with respect to deterministic (uniform p-time) observers, which was carried out by Goldreich and Wigderson [10]. Specifically, for every polynomial p, they presented a polynomial-time construction of a sample space that fools any p-time *deterministic next-bit test*. They observed that an analogous construction with respect to general (deterministic p-time) tests (i.e., distinguishers) would yield some non-trivial derandomization results (e.g., any unary set in $\mathcal{BPP}$ would be placed in $\mathcal{P}$). Thus, they concluded that there is a fundamental gap between probabilistic and deterministic polynomial-time observers. Our key observation is that this gap may disappear if $\mathcal{BPP} = \mathcal{P}$. Specifically, the hypothesis $\mathcal{BPP} = \mathcal{P}$ allows us to derandomize a trivial "probabilistic polynomial-time construction" of a canonical derandomizer.

Proof: Our starting point is the fact that, for some exponential function ℓ, with very high probability, a random function $G : \{0,1\}^k \to \{0,1\}^{\ell(k)}$ satisfies the pseudorandomness requirement associated with $2p$-robust canonical derandomizers. Furthermore, given the explicit description of any function $G : \{0,1\}^k \to \{0,1\}^{\ell(k)}$, we can efficiently distinguish between the case that G is $2p$-robust and the case that G is not p-robust.[17] Thus, the construction of a suitable pseudorandom generator is essentially a BPP-search problem. Next, applying Theorem 3.5, we can deterministically reduce this construction problem to $\mathcal{BPP}$. Finally, using the hypothesis $\mathcal{BPP} = \mathcal{P}$, we obtain a deterministic construction. Details follow.

[17] Formally, the asymptotic terminology of p-robustness is not adequate for discussing finite functions mapping k-bit long strings to $\ell(k)$-bit strings. However, as detailed below, what we mean is distinguishing (in probabilistic polynomial-time) between the case that G is "$2p$-robust" with respect to a given list of p-time machines and the case that G is not "p-robust" with respect to this list.

Let us fix an arbitrary polynomial p, and consider a suitable exponential function ℓ (to be determined later). Our aim is to construct a sequence of mappings $G : \{0,1\}^k \to \{0,1\}^{\ell(k)}$, for arbitrary $k \in \mathsf{N}$, that meets the requirements of a p-robust canonical derandomizer. It will be more convenient to construct a sequence of sets $S = \cup_{k\in\mathsf{N}} S_{\ell(k)}$ such that $S_n \subseteq \{0,1\}^n$, and let $G(i)$ be the i^{th} string in $S_{\ell(k)}$, where $i \in [2^k] \equiv \{0,1\}^k$. (Thus, the stretch function $\ell : \mathsf{N} \to \mathsf{N}$ satisfies $\ell(\log_2 |S_n|) = n$, whereas we shall have $|S_n| = \text{poly}(n)$, which implies $\ell(O(\log n)) = n$ and $\ell(k) = \exp(\Omega(k))$.) The set S_n should be constructed in $\text{poly}(n)$-time (so that G is computable in $\text{poly}(2^k \cdot \ell(k))$-time), and the pseudorandomness requirement of G coincides with requiring that, for every probabilistic p-time algorithm D, and all sufficiently large n, it holds that[18]

$$\left| \Pr[D(U_n)=1] - \frac{1}{|S_n|} \cdot \sum_{s\in S_n} \Pr[D(s)=1] \right| < \frac{1}{p(n)}. \tag{9}$$

Specifically, we consider an enumeration of (modified)[19] probabilistic p-time machines, and focus on fooling (for each n) the $p(n)$ first machines, where fooling a machine D means that Eq. (9) is satisfied (w.r.t this D). Note that, with overwhelmingly high probability, a random set S_n of size $K = \widetilde{O}(p(n)^2)$ satisfies Eq. (9) (w.r.t the $p(n)$ first machines). Thus, the following search problem, denoted $\mathtt{CON}^{(p)}$, is solvable in probabilistic $\widetilde{O}(p(n)^2 \cdot n)$-time: *On input 1^n, find a K-subset S_n of $\{0,1\}^n$ such that Eq. (9) holds for each of the $p(n)$ first machines.*

Next, consider the following promise problem $\mathtt{CC}^{(p)}$ (which is a companion of $\mathtt{CON}^{(p)}$). The *valid instance-solution pairs of* $\mathtt{CC}^{(p)}$ *are pairs* $(1^n, S_n)$ *such that for each of the first* $p(n)$ *machines* Eq. (9) *holds with* $p(n)$ *replaced by* $2p(n)$, and its *invalid instance-solution pairs are pairs* $(1^n, S_n)$ *such that for at least one of the first* $p(n)$ *machines* Eq. (9) *does not hold.* Note that $\mathtt{CC}^{(p)}$ is a BPP-search problem (as per Definition 3.2), and that it is indeed a companion of $\mathtt{CON}^{(p)}$ (as per Observation 3.3). Thus, by Theorem 3.5,[20] solving the search problem $\mathtt{CON}^{(p)}$ is deterministically (polynomial-time) reducible to some promise problem in $\mathcal{BPP}$. Finally, using the hypothesis $\mathcal{BPP} = \mathcal{P}$, the theorem follows. ∎

Observation 4.6 (on the exact complexity of the construction): *Note that* (by Theorem 3.5) *the foregoing reduction of* $\mathtt{CON}^{(p)}$ *to* $\mathcal{BPP}$ *runs in time* $t(n) = \widetilde{O}(p(n)^2 \cdot n)$*, whereas the reduction is to a problem in quartic-time, because the verification problem associated with* $\mathtt{CC}^{(p)}$ *is in sub-quadratic probabilistic time.*[21] *Thus, assuming that probabilistic quartic-time is in* $\text{DTIME}(p_4)$*, for some polynomial* p_4 (see Proposition 3.7)*, it follows that* $\mathtt{CON}^{(p)} \in \text{DTIME}(p')$*, where* $p'(n) = p_4(t(n))$.

Observation 4.7 (including the seed in the output sequence): *The construction of the generator* G (or the set S_n) *can be modified such that for every* $s \in \{0,1\}^k$ *the* k*-bit long prefix of* $G(s)$ *equals* s (i.e., the i^{th} string in S_n starts with the $(\log_2 |S_n|)$-bit long binary expansion of i).

[18] In [10, Thm. 2] the set S_n was only required to fool *deterministic* tests of the next-bit type.

[19] Recall that one cannot effectively enumerate all machines that run within some given time bound. Yet, one can enumerate all machines, and modify each machine in the enumeration such that the running-time of the modified machine respects the given time bound, while maintaining the functionality of the original machines in the case that the original machine respects the time bound. This is done by simply incorporating a time-out mechanism.

[20] See also the discussion just following the statement of Theorem 3.5, which asserts that if the search problem of a companion of Π is reducible to $\mathcal{BPP}$ then the same holds for Π.

[21] On input $(1^n, S)$ we need to compare the average performance of $p(n)$ machines on S versus their average performance on $\{0,1\}^n$, where each machine makes at most $p(n)$ steps. Recalling that $|S| = K = \widetilde{O}(p(n)^2)$, and that it suffices to get an approximation of the performance on $\{0,1\}^n$ up to an additive term of $1/2p(n)$, we conclude that the entire task can be performed in time $p(n) \cdot \widetilde{O}(p(n)^2) \cdot p(n) < (n + |S|n)^2$ (i.e., the number of machines times the number of experiments (which is $|S| + \widetilde{O}(p(n)^2)$) times the running time of one experiment).

Observation 4.7 implies that a (deterministic) polynomial-time distinguisher, which runs for more time than the foregoing generator, can distinguish the generator's output from a truely random sequence. Next, we show that, in certain cases, the distinguishing task is extremely easy (i.e., can be performed in sub-linear time) if the distinguisher is provided with an auxiliary input that can be generated in polynomial-time independently of the tested string.

A separation between Definition 4.2 and Definition 4.3: The p-robust canonical derandomizer constructed in the foregoing proof (or rather a small variant of it) does *not* satisfy the notion of a canonical derandomizer stated in Definition 4.2. Indeed, in this case, a (deterministic) polynomial-time finder F, which runs for more time than the foregoing generator, can find a string x that allows very fast distinguishing. Details follow.

The variant that we refer to is different from the one used in the proof of Theorem 4.5 only in the details of the underlying randomized construction. Instead of selecting a random set of $\widetilde{O}(p(n)^2)$ strings, we select $m = O(p(n)^3)$ strings in a pairwise independent manner. (This somewhat bigger set suffices to make the probabilistic argument used in the proof of Theorem 4.5 go through.) Furthermore, we consider a specific way of generating such an m-long sequence over $\{0,1\}^n$: For $b = \log_2 m$ and $t = n/b$, we generate an m-long sequence by selecting uniformly $(r_1, s_1), ..., (r_t, s_t) \in \{0,1\}^{2b}$, and letting the i^{th} string in the m-long sequence be the concatenation of the t strings $r_1 + i \cdot s_1$,..., $r_t + i \cdot s_t$ (where the arithmetics is of $\mathrm{GF}(2^b)$). (In the actual determintic construction of S_n (and G) a sutibale sequence $((r_1, s_1), ..., (r_t, s_t)) \in \{0,1\}^{2bt}$ is found and fixed, and the $G(i)$ equals the concatenation of the t strings $r_1 + i \cdot s_1$,..., $r_t + i \cdot s_t$.) Referring to this specific construction, we propose the following attack:

- The finder F determines the set S_n (just as the generator does). In particular, F *determines the elements* r_1, s_1, r_2, s_2 *used in its construction*, finds $\alpha, \beta \in \mathrm{GF}(2^b)$ such that $\alpha s_1 + \beta s_2 = 0$ and $(\alpha, \beta) \neq (0,0)$, lets $\gamma = \alpha \cdot r_1 + \beta \cdot r_2$, and encodes (α, β, γ) in the $3b$-bit long prefix of x.
- On input x (viewed as starting with the $3b$-bit long prefix $(\alpha, \beta, \gamma) \in \mathrm{GF}(2^b)^3$) and a tested n-bit long string that is viewed as a sequence $(z_1, ..., z_t) \in \mathrm{GF}(2^b)^t$, the distinguisher D *output 1 if and only if* $\alpha \cdot z_1 + \beta \cdot z_2 = \gamma$.

Note that $D(x, G(U_k))$ is identically 1 (because $\alpha \cdot (r_1 + i \cdot s_1) + \beta \cdot (r_2 + i \cdot s_2)$ equals $\gamma = \alpha \cdot r_1 + \beta \cdot r_2$ for every $i \in [m]$), whereas $\Pr[D(x, U_{\ell(k)}) = 1] = 2^{-b}$ (because a fixed non-zero linear combination of two random elements of $\mathrm{GF}(2^b)$ is uniformly distributed in $\mathrm{GF}(2^b)$).

Non-resilience to multiple samples. The foregoing example also demonstrates the non-resilience of Definition 4.3 to multiple samples. Specifically, consider a distinguisher D that obtains three samples, denoted $(z_1^{(1)}, ..., z_t^{(1)})$, $(z_1^{(2)}, ..., z_t^{(2)})$, and $(z_1^{(3)}, ..., z_t^{(3)})$ (each viewed as a t-long sequence over $\mathrm{GF}(2^b)$), and outputs 1 if and only if $(z_1^{(1)} - z_1^{(2)}) \cdot (z_2^{(2)} - z_2^{(3)}) = (z_2^{(1)} - z_2^{(2)}) \cdot (z_1^{(2)} - z_1^{(3)})$. Then, $D(G(i_1), G(i_2), G(i_3)) = 1$ for every $i_1, i_2, i_3 \in [2^k]$ (because $G(i_1)_j - G(i_2)_j = (i_1 - i_2) \cdot s_j$ and $G(i_2)_j - G(i_3)_j = (i_2 - i_3) \cdot s_j$ for every $j \in [t]$, which implies that each of the two compared products equals $(i_1 - i_2)(i_2 - i_3) \cdot s_1 s_2$), whereas $D(U_{\ell(k)}^{(1)}, U_{\ell(k)}^{(2)}, U_{\ell(k)}^{(3)})$ equals 1 with probability 2^{-b} (because the two compared products are uniformly distributed in $\mathrm{GF}(2^b)$ independently of one another).

4.3 A tedious tightening

Recall that we (kind of) showed that canonical derandomizers of exponential stretch imply that $\mathcal{BPP}$ is "effectively" contained in $\mathcal{P}$ (in the sense detailed in Definition 4.8), whereas $\mathcal{BPP} = \mathcal{P}$ implies the existence of the former. In this section we tighten this relationship by showing that the existence of canonical derandomizers of exponential stretch also follows from the hypothesis that $\mathcal{BPP}$ is "effectively" (rather than perfectly) contained in $\mathcal{P}$.

Definition 4.8 (effective containment): *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two classes of promise problems, and let $t : \mathsf{N} \to \mathsf{N}$. We say that $\mathcal{C}_1$ is t-effectively contained in $\mathcal{C}_2$ if for every $\Pi \in \mathcal{C}_1$ there exists $\Pi' \in \mathcal{C}_2$ such that for every probabilistic t-time algorithm F and all sufficiently large n it holds that $\Pr[F(1^n) \in \nabla(\Pi, \Pi') \cap \{0,1\}^n] < 1/t(n)$, where $\nabla(\Pi, \Pi')$ denotes the symmetric difference between $\Pi = (\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$ and $\Pi' = (\Pi'_{\mathrm{YES}}, \Pi'_{\mathrm{NO}})$* (i.e., $\nabla(\Pi, \Pi') \stackrel{\mathrm{def}}{=} \nabla(\Pi_{\mathrm{YES}}, \Pi'_{\mathrm{YES}}) \cup \nabla(\Pi_{\mathrm{NO}}, \Pi'_{\mathrm{NO}})$, where $\nabla(S, S') \stackrel{\mathrm{def}}{=} (S \setminus S') \cup (S' \setminus S)$).

Theorem 4.9 *The following two conditions are equivalent.*

1. *For every polynomial p, it holds that $\mathcal{BPP}$ is p-effectively contained in $\mathcal{P}$.*
2. *For every polynomial p, there exists a p-robust canonical derandomizer of exponential stretch.*

Proof: We first prove that Condition 2 implies Condition 1. (Indeed, this assertion was made several times in the foregoing discussions, and here we merely detail its proof.)

Let $\Pi = (\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$ be an arbitrary problem in $\mathcal{BPP}$, and consider the corresponding probabilistic linear-time algorithm A (of negligible error probability) derived for a padded version of Π, denoted $\Psi = (\Psi_{\mathrm{YES}}, \Psi_{\mathrm{NO}})$. Specifically, suppose that for some polynomial p_0, it holds that $\Psi_{\mathrm{YES}} = \{x0^{p_0(|x|)-|x|} : x \in \Pi_{\mathrm{YES}}\}$ and ditto for Ψ_{NO}. Now, for any polynomial p, consider the promise problem $\Psi' = (\Psi'_{\mathrm{YES}}, \Psi'_{\mathrm{NO}})$ such that

$$\Psi'_{\mathrm{YES}} \stackrel{\mathrm{def}}{=} \{x \in \Psi_{\mathrm{YES}} : \Pr[A_G(x) = 1] > 0.6\} \tag{10}$$

$$\Psi'_{\mathrm{NO}} \stackrel{\mathrm{def}}{=} \{x \in \Psi_{\mathrm{NO}} : \Pr[A_G(x) = 1] < 0.4\}, \tag{11}$$

where A_G is the algorithm obtained by combining A with a p-robust derandomizer G of exponential stretch ℓ (i.e., $A_G(x, s) = A(x, G(s))$, where $\ell(|s|) = |x|$). Then, Proposition 4.4 implies that for every probabilistic p-time algorithm F and all sufficiently large k, it holds that

$$\Pr[F(1^{\ell(k)}) \in \nabla(\Psi, \Psi') \cap \{0,1\}^{\ell(k)}] < \frac{40}{p(\ell(k))^{1/3}}, \tag{12}$$

because $\nabla(\Psi, \Psi') \cap \{0,1\}^{\ell(k)}$ is contained in $\nabla_{A,G}(k) \setminus B_A(k)$, where $\nabla_{A,G}(k)$ and $B_A(k)$ are as in Eq. (6) and Eq. (7), respectively. Now, since G has exponential stretch, it follows that the randomness complexity of A_G is logarithmic (in its input length). Thus, algorithm A_G runs in polynomial-time, and we can also fully derandomize it in polynomial-time (by invoking A_G on all possible random pads). Concluding that $\Psi' \in \mathcal{P}$, we further infer that the same holds with respect to the "unpadded version" of Ψ', denoted $\Pi' = (\Pi'_{\mathrm{YES}}, \Pi'_{\mathrm{NO}})$; that is, we refer to $\Pi'_{\mathrm{YES}} = \{x : x0^{p_0(|x|)-|x|} \in \Psi'_{\mathrm{YES}}\}$ and ditto for Π'_{NO}. Finally, since $\nabla(\Pi, \Pi') \cap \{0,1\}^n$ equals $\{x : x0^{p_0(|x|)-|x|} \in \nabla(\Psi, \Psi') \cap \{0,1\}^{p_0(n)}\}$, it follows that for every probabilistic $p \circ p_0$-time algorithm F and all sufficiently large n, it holds that $\Pr[F(1^n) \in \nabla(\Pi, \Pi') \cap \{0,1\}^n] < 40/p(p_0(n))^{1/3}$. Noting that the same applies to any $\Pi \in BPP$ (and any polynomial p), we conclude that $\mathcal{BPP}$

is $(p^{1/3}/40)$-effectively contained in $\mathcal{P}$, for every polynomial p. This completes the proof that Condition 2 implies Condition 1.

We now turn to proving the converse (i.e., that Condition 1 implies Condition 2). The idea is to go through the proof of Theorem 4.5, while noting that a failure of the resulting generator (which is supposed to be p-robust) yields contradiction to the p'-effective containment of $\mathcal{BPP}$ in $\mathcal{P}$, where p' is a polynomial that arises from the said proof. Specifically, note that the hypothesis $\mathcal{BPP} = \mathcal{P}$ is used in the proof of Theorem 4.5 to transform a probabilistic construction into a deterministic one. This transformation is actually a (deterministic) p^3-time[22] reduction (of the construction problem) to a fixed problem Π in $\text{BPTIME}(p_\Pi) \subseteq \mathcal{BPP}$, where $p_\Pi(m) = m^4$. We also note that all queries made by the reduction have length $\Theta(2^k \cdot \ell(k))$ (see the proof of Theorem 3.5, and recall that $2^k = \widetilde{O}(p(\ell(k))^2)$). Thus, the reduction fails only if at least one of the queries made by it is answered incorrectly by the problem in $\mathcal{P}$ that is used to p'-effective place Π in $\mathcal{P}$. Let us suppose for a moment that the reduction never makes a query that violates the promise of Π. Then, randomly guessing the index of the (first wrongly answered) query ($i \in [p(\ell(k))^3]$), we may answer the prior $(i-1)$ queries by using the fixed BPP algorithm for Π, and hit an m-bit long instance in the symmetric difference with probability at least $1/p(n)^3$, where $n = \ell(k)$ and $m = \widetilde{\Theta}(p(n)^2 \cdot n)$. Thus, for a sufficiently large polynomial p', this contradicts the hypothesis that $\mathcal{BPP}$ is p'-effectively contained in $\mathcal{P}$. Specifically, on input 1^n, our probabilistic algorithm runs for time $p(n)^3 \cdot p_\Pi(p(n)^3) = p(n)^{15}$ and hits a bad m-bit long string (on which the derandomization fails) with probability at least $1/p(n)^3$, where $m = \widetilde{\Theta}(p(n)^2 \cdot n)$. Thus, setting $p'(m) = m^8$ suffices. (Formally, the claim follows by considering a modified algorithm that on input 1^m invokes the foregoing algorithm on input $1^{m^{1/8}}$.)

Recall, however, that the foregoing analysis relies on the unjustified assumption that the reduction never makes a query that violates the promise of Π. In general, when such a query is made, the answer of the deterministic algorithm A (which p'-effectively places Π in $\mathcal{P}$) may be arbitrary and may not reflect the arbitrary distribution of the answer of the BPP algorithm, deboted B. By randomizing the reduction we may avoid this violation event (or rather bound the probability that it occurs), without effecting the behavior on queries that satisfy the promise. Before detailing how this is done, we stress that this modification will be performed only in the analysis, towards showing that failure of the original deterministic reduction when using algorithm A implies hitting a query on which A returns an incorrect answer. Turning back to the reduction to $\Pi = (\Pi_{\text{YES}}, \Pi_{\text{NO}})$ (which makes a number of queries that is smaller than the query length), we consider the problem $\Pi' = (\Pi'_{\text{YES}}, \Pi'_{\text{NO}})$ such that $(x,\alpha) \in \Pi'_{\text{YES}}$ (resp., $(x,\alpha) \in \Pi'_{\text{NO}}$) if and only if $\Pr[B(x)=1] > \alpha + 1/20|x|$ (resp., $\Pr[B(x)=1] < \alpha - 1/20|x|$). Clearly, $\Pi' \in \mathcal{BPP}$ (specifically, $\Pi' \in \text{BPTIME}(p_{\Pi'})$, where $p_{\Pi'}(m) = m^6$). In the reduction, we replace each query x by the query (x,α) such that α is selected at random uniformly in $[0.4, 0.6]$. Thus, for every x that satisfies the promise of Π and every $\alpha \in [0.4, 0.6]$, it holds that (x,α) satisfies the promise of Π'. On the other hand, with probability at least $1 - |x| \cdot (2 \cdot (20|x|)^{-1}/0.2) = 1/2$, all queries made by the reduction (to Π') satisfy the promise, since a query (x,α) violates the promise if and only if $|\Pr[B(x)=1] - \alpha| \leq 1/30|x|$. Now, let A' be a deterministic algorithm that p'-effectively places Π' in $\mathcal{P}$, and let $A(x) = A'(x, 0.5)$. (Indeed, we are using an algorithm derived from the algorithm for Π', rather than using the algorithm derived directly for Π.) Now, if A fails during the original deterministic reduction, then, with probability at least $2/3$, algorithm A' fails during the randomized reduction (i.e., answers some query incorrectly while all queries satisfy the promise). Hence,

[22] See Observation 4.6, and use $\widetilde{O}(p(n)^2 \cdot n) \ll p(n)^3$, which holds for all practical purposes.

we derive a contradiction to the hypothesis that Π' is p'-effectively in $\mathcal{P}$ (via algorithm A').[23] ■

Comment. The second part of the foregoing proof actually establishes that *there exists a fixed polynomial p' such that if $\mathcal{BPP}$ is p'-effectively contained in $\mathcal{P}$, then, for every every polynomial p, there exists a p-robust canonical derandomizer of exponential stretch.* Thus, we obtain that *$\mathcal{BPP}$ is p'-effectively contained in $\mathcal{P}$ if and only if for every polynomial p $\mathcal{BPP}$ is p-effectively contained in $\mathcal{P}$.* We comment that this result can be proved directly by a padding argument.

4.4 A different tightening (targeted generators)

The use of uniform-complexity notions of canonical derandomizers does not seem to allow deriving perfect derandomization (of the type $\mathcal{BPP} = \mathcal{P}$). As we saw, the problem is that exceptional inputs (in the symmetric difference between the original problem and the one solved deterministically) need to be found in order to yield a violation of the pseudorandomness condition. An alternative approach may let the generator depend on the input for which we wish to derandomize the execution of the original probabilistic polynomial-time algorithm. This suggests 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 4.10 (targeted canonical derandomizers): *Let $\ell\colon \mathsf{N} \to \mathsf{N}$ 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 $\mathrm{poly}(2^k \cdot \ell(k))$ 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*

$$|\Pr[D(x, G(U_k, x)) = 1] - \Pr[D(x, U_{\ell(k)}) = 1]| < \frac{1}{6}. \tag{13}$$

Definition 4.10 is a special case of related definitions that have appeared in [26, Sec. 2.4]. Specifically, Vadhan [26] studied auxiliary-input pseudorandom generators (of the general-purpose type [2, 27]), 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 $I \subseteq \{0,1\}^*$).[24] (On the other hand, Definition 4.1 is obatined from Definition 4.10 by mandating that G ignores s; i.e., $G(s,x) = G'(s)$.)

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 [20]. 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).

Theorem 4.11 (another equivalence): *Targeted canonical derandomizers of exponential stretch exist if and only if $\mathcal{BPP} = \mathcal{P}$.*

[23] Note that here we use $p'(m) = m^{11}$, since the running-time of our probabilistic process is $p(n)^3 \cdot p_{\Pi'}(p(n)^3) = p(n)^{21}$, where $m = \widetilde{\Theta}(p(n)^2 \cdot n)$.

[24] His treatment vastly extends the original notion of auxiliary-input one-way functions put forward in [21].

Proof: Using any targeted canonical derandomizer of exponential stretch we obtain $\mathcal{BPP} = \mathcal{P}$, where the argument merely follows the one used in the context of non-uniformly strong canonical derandomizers (i.e., canonical derandomizers in the sense of Definition 4.1). Turning to the opposite direction, we observe that the construction undertaken in the proof of Theorem 4.5 can be carried out with respect to the given auxiliary-input. In particular, the fixed auxiliary-input is merely passed among the various algorithms, and the argument remains intact. (See further discussion in Observation 4.12.) The theorem follows. ■

Observation 4.12 (super-exponential stretch): *In contrast to the situation with respect to the prior notions of canonical derandomizers* (of Definitions 4.1–4.3),[25] *targeted canonical derandomizer of super-exponential stretch may exist. Indeed, they exists if and only if targeted canonical derandomizer of exponential stretch exist. To see this note that the hypothesis $\mathcal{BPP} = \mathcal{P}$ allows to carry out the proof of Theorem 4.5 for any stretch function. Specifically, for any super-exponential function ℓ, when constructing the set $S_n \subset \{0,1\}^n$ it suffices to fool the first $g(n)$ (linear-time) machines, where g is any unbounded and non-decreasing function and fooling means keeping the distinguishability gap below $1/6$. Thus, $|S_n| = 2^{\ell^{-1}(n)}$ (which is $o(n)$) needs only satisfy $2 \cdot \exp(-2 \cdot (1/6)^2 \cdot |S_n|) \cdot g(n) < 1/3$, which calls for using a function g such that $g(n) \leq 0.1 \cdot \exp(2 \cdot (1/6)^2 \cdot 2^{\ell^{-1}(n)})$. The claim follows.*

4.5 Relating the various generators

It is syntactically clear that any non-uniformly strong canonical derandomizer (as per Definition 4.1) satisfies both Definition 4.2 (the first uniform version of canonical derandomizers) and Definition 4.10 (the targeted version of canonical derandomizers). On the other hand, there are good reasons to believe that such a canonical derandomizer is not necessarily a p-robust canonical derandomizer (as per Definition 4.3, for some polynomial p).[26] However, using Theorems 4.9 and 4.11, we observe that the existence of a generator that satisfies either Definition 4.2 or Definition 4.10 implies, for every polynomial p, the existence of p-robust canonical derandomizer (as per Definition 4.3).

Corollary 4.13 *If there exists a targeted canonical derandomizer of exponential stretch, then for every polynomial p there exists a p-robust canonical derandomizer of exponential stretch. The same holds if the hypothesis refers to Definition 4.2.*

The various relations are depicted in Figure 3. A similar result can be proved for other (polynomially closed) families of stretch functions, by using the results of Section 5.

Proof: The existence of a targeted canonical derandomizer of exponential stretch implies that $\mathcal{BPP} = \mathcal{P}$ (see Theorem 4.11), which in turn implies the existence of a p-robust canonical derandomizer of exponential stretch (see Theorem 4.5 or Theorem 4.9). Starting with a generator that satisfies Definition 4.2, one can easily prove that, for every polynomial p', it holds that $\mathcal{BPP}$ is p'-effectively in $\mathcal{P}$, where the proof is actually more direct than the corresponding direction of Theorem 4.9. We are done by using the other direction of Theorem 4.9 (i.e., the construction of p-robust canonical derandomizer based on p'-effective containment of $\mathcal{BPP}$ in $\mathcal{P}$). ■

[25] For Definitions 4.1 and 4.2 super-exponential stretch is impossible because we can encode in $x \in \{0,1\}^{\ell(k)}$ the list of all $(k+1)$-bit long strings that do not appear as a prefix of any string in $\{G(s) : s \in \{0,1\}^k\}$, which yields a linear-time distinguisher of gap at least $1/2$. In case of Definition 4.3, super-exponential stretch is impossible because of a distinguisher that output 1 if and only if the tested string starts with 0^{k+1}, and so has a distinguishing gap of at least $2^{-(k+1)}$. Indeed, in both cases we ruled out $\ell(k) \geq 2^{k+1}$.

[26] One such reason was noted in Footnote 15: If $\mathcal{E}$ requires exponential size circuits, then such a "separator" exists.

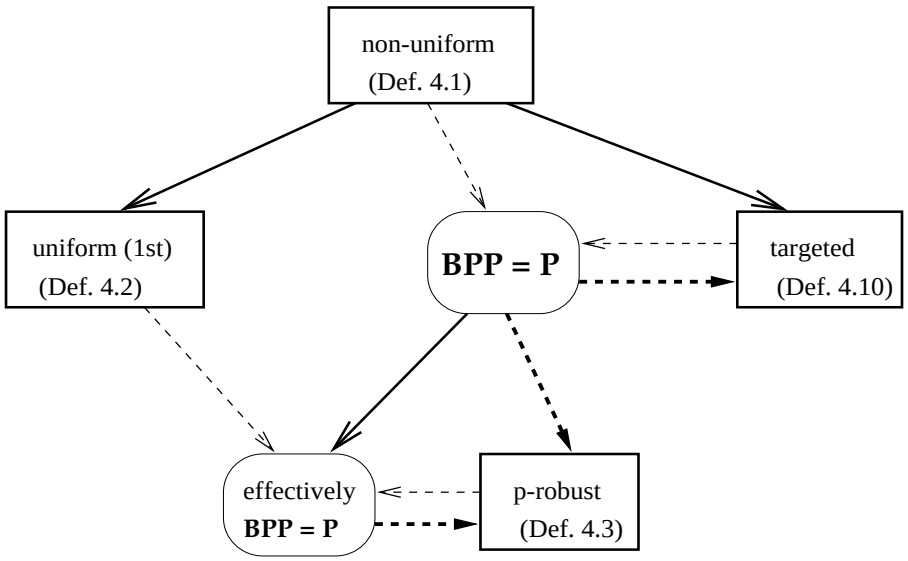

Figure 3: Relations among various notions of canonical derandomizers (of exponential stretch). Solid arrows indicate syntactic implications (which hold for any generator), whereas dashed arrows indicate existential implications.

5 Extension: the full "stretch vs time" trade-off

In this section we extend the ideas of the previous section to the study to general "stretch vs derandomization time" trade-off (akin to the general "hardness vs randomness" trade-off). That is, here the standard hardness vs randomness trade-off takes the form of a trade-off between the stretch function of the canonical derandomizer and time complexity of the deterministic class containing $\mathcal{BPP}$. The robustness (resp., effectiveness) function will also be adapted accordingly.

Theorem 5.1 (Theorem 4.9, generalized): *For every function $t:\mathsf{N}\to\mathsf{N}$, the following two conditions are equivalent.*

1. *For every two polynomials p_0 and p, it holds that* $\text{BPTIME}(p_0)$ *is $(p\circ t)$-effectively contained in* $\text{DTIME}(\text{poly}(p\circ t\circ p_0))$.
2. *For every polynomial p, there exists a $(p\circ t)$-robust canonical derandomizer of stretch $\ell_{p\circ t}:\mathsf{N}\to\mathsf{N}$ such that $\ell_{p\circ t}(k)\stackrel{\text{def}}{=}(p\circ t)^{-1}(2^{\Omega(k)})=t^{-1}(p^{-1}(2^{\Omega(k)}))$.*

Furthermore, the hidden constants in the Ω and poly *notation are independent of the functions t,p and p_0.*

Indeed, Theorem 4.9 follows as a special case (when setting $t(n)=n$), whereas for $t(n)\geq 2^n$ both conditions hold trivially. Note that for $t(n)=2^{\epsilon n}$ (resp., $t(n)=2^{n^\epsilon}$), we get $\ell_{p\circ t}(k)=\Omega(k/\epsilon)$) (resp., $\ell_{p\circ t}(k)=\Omega(k)^{1/\epsilon}$).

Proof: We closely follow the proof of Theorem 4.9, while detailing only the necessary modifications. Starting with the proof that Condition 2 implies Condition 1, we let $\Pi\in\text{BPTIME}(p_0)$, Ψ and A be as in the original proof. Now, for any polynomial p, we consider the promise problem $\Psi'=(\Psi'_{\text{YES}},\Psi'_{\text{NO}})$ such that $\Psi'_{\text{YES}}=\{x\in\Psi_{\text{YES}}:\Pr[A_G(x)=1]>0.6\}$ and $\Psi'_{\text{NO}}=\{x\in\Psi_{\text{NO}}:\Pr[A_G(x)=1]<0.4\}$, where A_G is the algorithm obtained by combining A with a $(p\circ t)$-robust derandomizer G of stretch

$\ell_{p\circ t}$. Then, Proposition 4.4 implies that for every probabilistic $(p \circ t)$-time algorithm F and all sufficiently large k, it holds that $\Pr[F(1^{\ell(k)}) \in \nabla(\Psi, \Psi') \cap \{0,1\}^{\ell(k)}] < 40/(p \circ t)^{1/3}(\ell(k))$. Since G has stretch $\ell_{p\circ t}$, it follows that on input an n-bit string algorithm A_G uses $\ell^{-1}_{p\circ t}(n) = O(\log(p\circ t)(n))$ many coins, and thus we can also fully derandomize it in time $\mathrm{poly}((p \circ t)(n))$. Thus, $\Psi' \in \text{DTIME}(\mathrm{poly}(p\circ t))$, and it follows that $\Pi' \in \text{DTIME}(\mathrm{poly}(p\circ t\circ p_0))$, where Π' denotes the "unpadded version" of Ψ'. Concluding that Π is $((p \circ t)^{1/3}/40)$-effectively contained in $\text{DTIME}(\mathrm{poly}(p \circ t \circ p_0))$, and that the same holds for any $\Pi \in \text{BPTIME}(p_0)$ and every polynomial p, we have established that Condition 2 implies Condition 1.

We now turn to proving the converse (i.e., that Condition 1 implies Condition 2). Again, we merely go through the proof of Theorem 4.5, except that here we construct a set S_n of size $\mathrm{poly}(p\circ t)(n)$. Specifically, the discrepancies we aim at are linearly related to $1/(p\circ t)(n)$, and we can afford spending time $\mathrm{poly}(p \circ t)(n)$ in the construction. We shall indeed make use of this allowance, since we can only rely on the (t'-effective) containment of $\text{BPTIME}(p_0)$ in $\text{DTIME}(\mathrm{poly}(p \circ t \circ p_0))$, where $t' = \mathrm{poly}(p \circ t) = \mathrm{poly}(p) \circ t$. The rest of the argument proceeds analogously to the proof of Theorem 4.9. We note that the aforementioned hypothesis regarding $\text{BPTIME}(p_0)$ is only used when deterministically reducing (in time $\mathrm{poly}(p \circ t)$) the construction of S_n to a fixed problem Π in $\text{BPTIME}(p_0)$, where $p_0(m) = m^4$ (as in the proof of Theorem 4.9). Thus, the reduction fails only if at least one of the queries made by it is answered incorrectly by the problem in $\mathcal{D} \stackrel{\text{def}}{=} \text{DTIME}(\mathrm{poly}(p \circ t \circ p_0))$ that is used to t'-effective place Π in $\mathcal{D}$. Randomly guessing the the index of the (wrongly answered) query, we hit an m-bit long instance in the symmetric difference with probability at least $1/\mathrm{poly}(p(t(n)))$, where $m = \Omega(\ell(k))$, which contradicts the hypothesis that $\text{BPTIME}(p_0)$ is t'-effectively contained in $\mathcal{D}$.[27] ■

6 Open Problems

We start by recalling the famous open problem regarding whether the a full derandomization of standard decision problems implies the same for promise problems. That is, assuming that any decision problem in $\mathcal{BPP}$ is in $\mathcal{P}$, does it follow that $\mathcal{BPP} = \mathcal{P}$?[28]

One problem that arises from the current work refers to the relationship between the two uniform definitions of canonical derandomizers (i.e., Definitions 4.2 and 4.3). Recall (see Section 4.5) that the existence of generators (of exponential stretch) that satisfy Definition 4.2 implies the existence of generators (of exponential stretch) that satisfy Definition 4.3, but the converse is not clear.

Another open problem refers to the deriving of analogous results regarding the derandomization of $\mathcal{AM}$ (or $\mathcal{AM} \cap \mathrm{co}\mathcal{AM}$). Here the canonical derandomizer should be computable in *non-deterministic* $\mathrm{poly}(2^k \cdot \ell(k))$-time, where computation by non-deterministic machines refers to the so called "single-value" model (see, e.g., [23] or [9, Def. 5.13]). The problem in reversing the "pseudorandomness to derandomization" connection refers to a tension between the distinguishers used to argue about the derandomization versus our need to handle them in the construction of the canonical derandomizer. We would welcome any result, even for a targeted version and even for derandomizing some subclass such as $\mathcal{AM} \cap \mathrm{co}\mathcal{AM}$ or $\mathcal{SZK}$.

Finally, we return to the question raised in passing in Section 1.4. Specifically, we ask *which search problems can be solved by deterministic polynomial-time reductions to $\mathcal{BPP}$*. Denoting the class of such search problems by $\mathcal{C}$, we note that Theorem 3.5 implies that $\mathcal{C}$ contains all search

[27] Here, too, $m = \Theta(\widetilde{O}(p(t(n))^2 \cdot n)$ actually holds, and so it actually suffices to set $t'(m) = \mathrm{poly}(m)$.

[28] Formally, let $\mathcal{D}$ denote the set of all promise problems having a trivial promise; that is, a promise problem $(\Pi_{\text{YES}}, \Pi_{\text{NO}})$ is in $\mathcal{D}$ if $\Pi_{\text{YES}} \cup \Pi_{\text{NO}} = \{0,1\}^*$. Then, the question is whether $\mathcal{BPP} \cap \mathcal{D} = \mathcal{P} \cap \mathcal{D}$ implies $\mathcal{BPP} = \mathcal{P}$.

problems that have a companion that is a BPP-search problem. The converse holds in the special case that the target of the reduction is a standard decision problem (and the reduced search problem has a trivial promise at the instance level (see below)). Let us consider the general case and see what happens. Suppose that the search problem $(R_{\text{YES}}, R_{\text{NO}})$ is reducible in deterministic polynomial-time to $\Pi \in \mathcal{BPP}$. Denoting the oracle machine effecting the reduction by M, we consider the search problem $(R'_{\text{YES}}, R'_{\text{NO}})$ such that $(x,y) \in R'_{\text{YES}}$ if $M^f(x) = y$ for some f that is consistent with Π and $(x,y) \in R'_{\text{NO}}$ otherwise.[29] The correctness of the reduction implies that $S_{R'_{\text{YES}}} \supseteq S_{R_{\text{YES}}}$ whereas $R'_{\text{NO}} \supseteq R_{\text{NO}}$, which means that if $S_{R_{\text{YES}}} \cup S_{R_{\text{NO}}} = \{0,1\}^*$, then $(R'_{\text{YES}}, R'_{\text{NO}})$ is a companion of $(R_{\text{YES}}, R_{\text{NO}})$. Now if Π is a standard decision problem, then f is unique; hence, $R'_{\text{YES}}(x)$ is a singleton if $x \in S_{R'_{\text{YES}}}$ and is empty otherwise (since $S_{R'_{\text{NO}}} = \{0,1\}^* \setminus S_{R'_{\text{YES}}}$). In this case membership of (x,y) in R'_{YES} can be easily tested by checking whether $M^{\Pi}(x) = y$. The same holds if the reduction is "smart" (i.e., avoids making queries that violate the promise, cf. [12]),[30] but in general it is not clear what happens.

Acknowledgments

We are grateful to Noga Alon, Or Meir, Madhu Sudan, and Avi Wigderson for useful discussions on related issues. We also thank Or Meir, Dieter van Melkebeek, Amnon Ta-Shma, and Salil Vadhan for their comments on early drafts of this work.

[29] Saying that f is consistent with $\Pi = (\Pi_{\text{YES}}, \Pi_{\text{NO}})$ means that $f(x) = 1$ for every $x \in \Pi_{\text{YES}}$, whereas $f(x) = 0$ for every $x \in \Pi_{\text{NO}}$. Indeed, the value of f on inputs that violate the promise (i.e., $x \notin \Pi_{\text{YES}} \cup \Pi_{\text{NO}}$) is arbitrary.

[30] We note, however, that the reduction used in the proof of Theorem 3.5 is not smart. Furthermore, we doudt that a smart reduction can be found.

References

[1] S. Aaronson, B. Aydinlioglu, H. Buhrman, D. Gutfreund, J.M. Hitchcock, A. Kawachi, and D. van Melkebeek. Derandomizing Arthur-Merlin Games and Approximate Counting Implies Exponential-Size Lower Bounds. *Computational Complexity*, to appear.

[2] 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.

[3] B. Chor and O. Goldreich. On the Power of Two–Point Based Sampling. *Jour. of Complexity*, Vol 5, 1989, pages 96–106.

[4] 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.

[5] L. Fortnow. Comparing Notions of Full Derandomization. In *16th CCC*, pages 28–34, 2001.

[6] J. Friedman. A Proof of Alon's Second Eigenvalue Conjecture. In *35th STOC*, pages 720–724, 2003.

[7] C.F. Gauss. *Untersuchungen Über Höhere Arithmetik*. Chelsea publishing company, second edition, reprinted, New York, 1981.

[8] O. Goldreich. *Foundation of Cryptography: Basic Tools*. Cambridge University Press, 2001.

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

[10] O. Goldreich and A. Wigderson, On Pseudorandomness with respect to Deterministic Observers. In *RANDOM'00, proceedings of the satellite workshops of the 27th ICALP*. Carleton Scientific (Proc. in Inform. 8), pages 77–84, 2000. See also *ECCC*, TR00-056.

[11] S. Goldwasser and S. Micali. Probabilistic Encryption. *JCSS*, Vol. 28, No. 2, pages 270–299, 1984. Preliminary version in *14th STOC*, 1982.

[12] J. Grollmann and A.L. Selman. Complexity Measures for Public-Key Cryptosystems. *SICOMP*, Vol. 17 (2), pages 309–335, 1988.

[13] D. Hochbaum (ed.). *Approximation Algorithms for NP-Hard Problems*. PWS, 1996.

[14] M.N. Huxley. On the Difference Between Consecutive Primes. *Invent. Math.*, Vol. 15, pages 164–170, 1972.

[15] R. Impagliazzo, V. Kabanets, and A. Wigderson. In Search of an Easy Witness: Exponential Time vs Probabilistic Polynomial Time. *JCSS*, Vol. 65 (4), pages 672–694, 2002. Preliminary version in *16th CCC*, 2001.

[16] R. Impagliazzo and A. Wigderson. P=BPP if E requires exponential circuits: Derandomizing the XOR Lemma. In *29th STOC*, pages 220–229, 1997.

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

[18] M. Jerrum, L. Valiant, and V.V. Vazirani. Random Generation of Combinatorial Structures from a Uniform Distribution. *TCS*, Vol. 43, pages 169–188, 1986.

[19] V. Kabanets and R. Impagliazzo. Derandomizing Polynomial Identity Tests Means Proving Circuit Lower Bounds. *Computational Complexity*, Vol. 13, pages 1-46, 2004. Preliminary version in *35th STOC*, 2003.

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

[21] 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.

[22] O. Reingold, L. Trevisan, and S. Vadhan. Pseudorandom walks on regular digraphs and the RL vs. L problem. In *38th STOC*, pages 457–466, 2006. See details in *ECCC*, TR05-022.

[23] R. Shaltiel and C. Umans. Low-end Uniform Hardness vs Randomness Tradeoffs for AM. *SICOMP*, Vol. 39, No. 3, pages 1006–1037, 2009. Preliminary version in *39th STOC*, 2007.

[24] L. Trevisan and S. Vadhan. Pseudorandomness and Average-Case Complexity Via Uniform Reductions. *Computational Complexity*, Vol. 16 (4), pages 331–364, 2007. Preliminary version in *17th CCC*, 2002.

[25] C. Umans. Pseudo-random Generators for all Hardness. *JCSS*, Vol. 67 (2), pages 419–440, 2003. Preliminary version in *34th STOC*, 2002.

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

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

Appendices: prior proofs of the main result (Theorem 4.5)

The current proof of Theorem 4.5 is the second simplification we found: It is a third incarnation of the same underlying principles, but it hides the original inspiration to our ideas, which are rooted in [10]. Since we do have written records of these prior proofs, and since they may be of some interest, we decided to include them in the current appendix.

Our starting point was the work of Goldreich and Wigderson [10], which studied pseudorandomness with respect to (uniform) deterministic observers. In particular, they show how to construct, for every polynomial p, a generator of exponential stretch that works in time polynomial in its output and fools all deterministic p-time tests of the next-bit type (a la [2]). They observe that an analogous construction with respect to general (deterministic p-time) tests (or distinguishers) would yield some non-trivial derandomization results (e.g., any unary set in $\mathcal{BPP}$ would be placed in $\mathcal{P}$). Thus, they conclude that Yao's result[31] asserting that *fooling all efficient next-bit tests implies fooling all efficient distinguishers* relies on the fact that the class of test includes probabilistic p-time algorithms and not only deterministic ones.

Our key observation is that the gap between probabilistic next-bit tests and deterministic ones essentially disappears if $\mathcal{BPP} = \mathcal{P}$. Actually, the gap disappears if we generalize the notion of next-bit tests so to allow the (deterministic) tester to output a guess of the probability that the next bit equals 1 (rather than a guess for the actual value of the next bit), and consider the correlation between the corresponding random variables. Indeed, assuming that $\mathcal{BPP} = \mathcal{P}$, allows to deterministically emulate a probabilistic p-time next bit test by a (generalized) deterministic p'-time next bit test, where p' is a polynomial that depends only on p. Plugging this into the construction of [10], which can be shown to fool also (generalized) deterministic p'-time next bit test, we obtain the desired generator (which produces ℓ-bit outputs in time $\mathrm{poly}(p'(\ell))$). A crucial point in the foregoing argument is that the next-bit test does not need to invoke the generator, which is not feasible because the generator runs for more time than the potential tests.

The foregoing argument led to the first proof, which is presented in Appendix A.2. Subsequently we found a more direct approach, which is presented in Appendix A.1. This approach is more transparent and amenable to variations than the first one (but less so in comparison to the proof presented in Section 4.2). Specifically, rather than working with (generalized) next-bit tests, we directly work with (probabilistic p-time) distinguishers, and *adapt the argument of [10] to apply in this context*. It turns out that in order for this to work, we only need to approximate the probability that a fixed probabilistic p-time distinguishers outputs 1 when presented with random $(\ell - i)$-bit long extensions of some fixed i-bit long strings, for $i = 1, ..., \ell$. Assuming that $\mathcal{BPP} = \mathcal{P}$, allows to deterministically approximate these probabilities (again, in p'-time, where $p' = \mathrm{poly}(p)$), and so we are done. Needless to say, the fact that such approximations suffices is specific to (our adaptation of) the construction of [10].

A.1 An alternative proof of Theorem 4.5 (via derandomizing a FPTAS)

The alternative proof of Theorem 4.5 proceeds by generalizing the main idea that underlies the work of Goldreich and Wigderson [10], while using the hypothesis (i.e., $\mathcal{BPP} = \mathcal{P}$) to extend its scope to probabilistic (rather than deterministic) observers. Specifically, for every polynomial p, they presented a polynomial-time construction of a sample space that fools any p-time *deterministic next-bit test*. The construction is iterative, where in each iteration the next bit of each string in the sample space is determined such that the resulting space fools all relevant next-bit tests. Here

[31] Attributed to oral presentations of [27].

we consider any (p-time) *probabilistic distinguisher*, and seek to determine the next bit so that the probability that this distinguisher output 1 (on a random extension of the current sample space) is approximately maintained. Towards this end, we need to approximate the probability that a fixed p-time *probabilistic* algorithm outputs 1 on a *random* extension of the current prefix. Our key observation is that, due to the hypothesis that $\mathcal{BPP} = \mathcal{P}$, this quantity can be approximated in deterministic polynomial-time. The use of this hypothesis is far from being surprising, since (as noted before) the conclusion of Theorem 4.5 implies that, in some "effective" sense, $\mathcal{BPP}$ does equal $\mathcal{P}$.

Proof: We follow the general outline of the proof of [10, Thm. 2], while commenting (mostly in footnotes) about the points of deviation. Let us fix an arbitrary polynomial p, and consider a suitable exponential function ℓ (to be determined later). Our aim is to construct a sequence of mappings $G : \{0,1\}^k \to \{0,1\}^{\ell(k)}$, for arbitrary $k \in \mathsf{N}$, that meets the requirements of a p-robust canonical derandomizer. However, it will be more convenient to construct a sequence of sets $S = \cup_{k\in\mathsf{N}} S_{\ell(k)}$ such that $S_n \subseteq \{0,1\}^n$, and let $G(i)$ be the i^{th} string in $S_{\ell(k)}$, where $i \in [2^k] \equiv \{0,1\}^k$. (Thus, the function $\ell : \mathsf{N} \to \mathsf{N}$ satisfies $\ell(\log_2 |S_n|) = n$, whereas we shall have $|S_n| = \text{poly}(n)$.) The set S_n should be constructed in poly(n)-time (so that G is computable in poly($2^k \cdot \ell(k)$)-time), and the pseudorandomness requirement of G coincides with requiring that, for every probabilistic p-time algorithm D, and all sufficiently large n, it holds that[32]

$$\left| \Pr[D(U_n)=1] - \frac{1}{|S_n|} \cdot \sum_{s\in S_n} \Pr[D(s)=1] \right| < \frac{1}{p(n)}. \tag{14}$$

Specifically, we consider an enumeration of (modified)[33] probabilistic machines running within time $p(n)$ on input of length n, and focus on fooling the $m = m(n) < p(n)$ first machines in the sense of Eq. (14). Let $\epsilon = 1/p(n)$, and M be a generic machines that we wish to fool.

We construct S_n in (roughly) n iterations, such that in iteration i we construct $S_{n,i} \subseteq \{0,1\}^i$. We start with $S_{n,k} = \{0,1\}^k$, where $k = 2\log_2(2nm/\epsilon)$, and let $K = 2^k$. In the $i+1^{\text{st}}$ iteration, we consider the function $f_M : [K] \to [0,1]$ representing the probability that M outputs 1 on a random extension of each of the K strings in $S_{n,i}$; that is, $f_M(j) = \Pr[M(x^{(j)}U_{n-i}) = 1]$, where $x^{(j)}$ is the j^{th} string in $S_{n,i} \subseteq \{0,1\}^i$. (The function f_M represents M's average output on all possible $(n-i)$-bit long extensions of all strings in $S_{n,i}$.)[34] Our aim is to find a vector $u \in \{0,1\}^K$ such that, for each machine M (among the first m machines), it holds that the average value of $\Pr[M(x^{(j)}u[j]U_{n-i-1})=1]$ is ϵ/n-close to the average value of $f_M(j)$; that is,

$$\left| \frac{1}{K} \sum_{j\in[K]} \Pr[M(x^{(j)}u[j]U_{n-i-1})=1] - \frac{1}{K} \cdot \sum_{j\in[K]} f_M(j) \right| < \frac{\epsilon}{n}. \tag{15}$$

Once such a vector u is found, we extend $S_{n,i}$ into $S_{n,i+1}$ in the natural manner; that is,

$$S_{n,i+1} \stackrel{\text{def}}{=} \{x^{(j)}u[j] : \text{ where } x^{(j)} \text{ is the } j^{\text{th}} \text{ string in } S_{n,i}\} \subset \{0,1\}^{i+1}. \tag{16}$$

[32] In [10, Thm. 2] the set S_n was only required to fool *deterministic* tests of the next-bit type.

[33] Recall that one cannot effectively enumerate all machines that run within some given time bound. Yet, one can enumerate all machines, and modify each machine in the enumeration such that the running-time of the modified machine respects the given time bound, while maintaining the functionality of the original machines in the case that the original machine respects the time bound. This is done by simply incorporating a time-out mechanism.

[34] In contrast, in [10], the function f_M (which is denoted v_M there) represented M's attempt to guess the $i+1^{\text{st}}$ bit of a string in S_n, based on the i-bit long prefix of that string. Furthermore, since in [10] the machine M is deterministic, the function f_M (there) can be constructed by invoking M on K different i-bit strings.

It follows that $S_n \stackrel{\text{def}}{=} S_{n,n}$ satisfies Eq. (14), because, for each of the aforementioned M's and for each $i \in [n-k, n-1]$, it holds that

$$\left| \frac{1}{|S_{n,i}|} \cdot \sum_{s \in S_{n,i}} \Pr[M(s)=1] - \frac{1}{|S_{n,i+1}|} \cdot \sum_{s \in S_{n,i+1}} \Pr[M(s)=1] \right| < \frac{\epsilon}{n} \tag{17}$$

since the terms in the l.h.s are represented by the function f_M defined at the i^{th} iteration, whereas the terms in the r.h.s correspond to the function f_M defined in the next iteration.

It remains to specify how a suitable vector $u \in \{0,1\}^K$ is found, in each iteration. This is done by using a pairwise independent sample space for strings of length K, while recalling that such spaces can be constructed in poly(K)-time (cf., e.g., [3]). Two issues arise:

1. Showing that such a sample space must always contain a suitable vector $u \in \{0,1\}^K$; that is, a vector $u \in \{0,1\}^K$ satisfies Eq. (15). This is quite an immediate consequence of the fact that, when defining $q_{M,j}(\sigma) \stackrel{\text{def}}{=} \Pr[M(x^{(j)}\sigma U_{n-i-1})=1]$, we can write Eq. (15) as

$$\left| \frac{1}{K} \sum_{j \in [K]} q_{M,j}(u[j]) - \frac{1}{K} \cdot \sum_{j \in [K]} \sum_{\sigma \in \{0,1\}} \frac{q_{M,j}(\sigma)}{2} \right| < \frac{\epsilon}{n}. \tag{18}$$

   Indeed, when u is selected from a pairwise-independent sample space of $\{0,1\}^K$, Eq. (18) holds with probability at least $1 - (1/((\epsilon/n)^2 K))$, and the claim follows whenever $(\epsilon/n)^2 K > m$.

   Actually, we shall use the fact that, with probability at least $1 - (1/((\epsilon/2n)^2 K))$, a modified version of Eq. (18) holds, where in the modification the upper bound ϵ/n is replaced by the (tighter) upper bound $\epsilon/2n$.

2. Showing that we can distinguish in deterministic polynomial-time whether a given vector $u \in \{0,1\}^K$ satisfies the aforementioned tighter form of Eq. (18) or violates Eq. (15).

   This issue hardly arose in [10], since there $f_M(j)$ referred to the output of a deterministic machine on fixed string (i.e., the j^{th} string in $S_{n,i}$). But here $f_M(j)$ refers to the expected output of a probabilistic machine on a random extension of a fixed string. Nevertheless, $f_M(j)$ can be approximated to within an additive term of $\epsilon/4n$ by a simple probabilistic algorithm that merely invokes M sufficiently many (i.e., $O(n/\epsilon)^2$) times on such random extensions, and ditto for $q_{M,j}(\cdot)$. Now, using the hypothesis $\mathcal{BPP} = \mathcal{P}$ and applying Corollary 3.6,[35] we conclude that $q_{M,j}(\cdot)$ can be approximated well-enough (i.e., up to an additive term of $\epsilon/4n$) by a deterministic polynomial-time algorithm. Formally, the approximation problem is defined for inputs of the form $(M, 1^n, x)$, where M and n are as above and x is a string of length at most n (i.e., in our application, $x = x^{(j)}\sigma \in \{0,1\}^{i+1}$, where $x^{(j)}$ is the j^{th} string in $S_{n,i}$).

Thus, in each iteration we can find a vector u as desired, an consequently we construct the desired set S_n in time that is polynomial in n. The theorem follows. ■

[35] Formally, there is a problem here since we do not have a FPTAS for the quantities $q_{M,j}(\cdot) \in [0,1]$, but we do have a FPTAS for the quantities $1 + q_{M,j}(\cdot) \in [1,2]$ and this suffices here.

Comment: The generator constructed in the foregoing proof does not satisfy the notion of a canonical derandomizer stated in Definition 4.2. Indeed, in this case, a (deterministic) polynomial-time finder F, which runs for more time than the foregoing generator, can find a string x that allows very fast distinguishing. Details follow.

Referring to the foregoing construction of a pseudorandom generator G, we show how to find in polynomial-time a string $x \in \{0,1\}^{\ell(k)}$ such that $(x, G(U_k))$ and $(x, U_{\ell(k)})$ are easy to tell apart. Actually, we refer to a specific implementation of the construction; that is, to a specific implementation of the pairwise independence sample space, where the aspect we rely on is that this sample space is a vector space of low dimension.

Recall that the set S_n, which is the support of $G(U_k)$ (for $n = \ell(k)$), is constructed by concatenating n vectors, where each vector is an 2^k-bit long sequence taken from a pairwise independence sample space. Specifically, consider the sample space generated by strings of length $k+1$ such that the j^{th} coordinate of the vector generated by $r_1 \cdots r_{k+1} \in \{0,1\}^{k+1}$ equals $\sum_{h=1}^{k+1} j_h r_h$, where j_h is the h^{th} bit in the $(k+1)$-bit long binary expansion of $j \in [2^k]$. Thus, each of the vectors used in the construction of S_n reside in the very same $(k+1)$-dimensional vector space.

Then, the finder F can just construct S_n (as done by G), and record the sequence $u_{k+1}, ..., u_n$ of vectors taken in each of the $n-k$ iterations (or rather their succinct representation). In fact, it suffices to record the indices of a subsequence, of length $d \leq k+2$, that sums-up to the all zero vector; that is, record $(i_1, ..., i_d)$ such that the sum of the vectors taken in iterations $i_1, ..., i_d$ equals the all-zero vector (i.e., $\sum_{h=1}^{d} u_{i_h} = 0$). Now, F just records such a sequence in a prefix of x, and the distinguisher D just outputs the XOR value of the bits that appear in these positions. Clearly, $D(x,z) = 0$ for every $z \in S_n$, whereas $\Pr[D(x, U_\ell) = 0] = 1/2$.

A.2 Another alternative proof of Theorem 4.5 (via next-bit tests)

The alternative proof of Theorem 4.5 follows by combining two results: The first is an unconditional result that asserts a generator that passes the next-bit test *with respect to deterministic observers*, and the second is a conditional *deterministic analogue* of the fact that next-bit unpredictability implies pseudorandomness. The first result is a technical generalization of [10, Thm. 2] (see Theorem A.2 below), whereas the second result uses the hypothesis that $\mathcal{BPP} = \mathcal{P}$. The use of this hypothesis should not come at a surprise, since as noted before the conclusion of Theorem 4.5 implies that in some "effective" sense $\mathcal{BPP}$ does equal $\mathcal{P}$. (Thus, as observed in [10], an unconditional implication of the foregoing type, would yield an unconditional derandomization of BPP.)

Next-bit unpredictability, generalized. Let us first present a natural generalization of the notion of next-bit unpredictability, which is pivotal to our argument. Recall that the standard notion refers to the probability of guessing the next bit in a sequence, when given its prefix. That is, in the standard formulation, the predictor outputs a bit, and the question is whether this bit matches the actual next bit. Now, consider a generalization in which the predictor output its estimate of the probability that the next bit is 1, and normalizing the "payoff" accordingly. That is, the generalized predictor outputs a probability $p \in [0,1]$, and gets payoff $2p-1$ if the outcome (of the next bit) is 1 and payoff $1-2p$ if the outcome is 0. (In general, the payoff is $(1-2p) \cdot (-1)^\sigma$, where σ denotes the outcome of the next bit.) Note that such a generalization allows the capture the intuitive case that the predictor wishes to pass on the guess, collect no potential gain but also suffer from no potential loss.

In the context of *probabilistic* algorithms, nothing is gained by this generalization, because a a probabilistic predictor may maintain the expected payoff by replacing the output $p \in [0,1]$ with a

biased coin toss (i.e., a random Boolean value that is biased with probability p towards 1).[36] But this generalization seems more powerful in the context of deterministic predictors, and we shall thus refer to it.

It will be more convenient to replace prediction probabilities with correlation, as we have already done implicitly above. Thus, the predictors will output a value in $v \in [-1,+1]$ (rather than in $[0,1]$) and collect the payoff $v \cdot (-1)^\sigma$, where σ denotes the outcome of the next bit. Thus, v corresponds to $1-2p$ in the forgoing discussion, and $v \cdot (-1)^\sigma$ corresponds to the correlation of v with $(-1)^\sigma$.

Definition A.1 (next bit unpredictability, generalized): *For $\ell : \mathsf{N} \to \mathsf{N}$, let $G : \{0,1\}^* \to \{0,1\}^*$ be such that for every $s \in \{0,1\}^*$ it holds that $|G(s)| = \ell(|s|)$.*

- *For $\epsilon : \mathsf{N} \to [0,1]$, we say that $A : \mathsf{N} \times \{0,1\}^* \to [-1,+1]$* has correlation at most ϵ with the next bit of G *if for all sufficiently large k and all $i < \ell(k)$, it holds that*

$$\mathrm{E}[A(1^{\ell(k)}, x_1, ..., x_i) \cdot (-1)^{x_{i+1}}] \leq \epsilon(\ell(k)), \tag{19}$$

  where $(x_1, ..., x_{\ell(k)}) \leftarrow G(s)$ for a uniformly selected $s \in \{0,1\}^k$.

  We will often omit $1^{\ell(k)}$ from the list of inputs to A.

- *We say that G is* next-bit unpredictable by a class of algorithms $\mathcal{A}$ with respect to an advantage ϵ *if every algorithm in $\mathcal{A}$ has correlation at most ϵ with the next bit of G.*

  We say that G is next-bit t-unpredictable *if G is next-bit unpredictable with respect to advantage $1/t$ by the class of deterministic t-time algorithms that have output in $[-1,+1]$.*

By a straightforward extension of the ideas of [10], we obtain the following unconditional result.

Theorem A.2 ([10, Thm. 2], generalized): *For every polynomial p, there exist an exponential function ℓ and a deterministic algorithm G that satisfies the first condition of Definition 4.3 such that G is next-bit p-unpredictable.*

The original result was stated in terms of predicting probabilities, and corresponds to the special case in which the algorithm's output is always in $\{-1,+1\}$ (where, in this case, $\mathrm{E}[A(x_1, ..., x_i) \cdot (-1)^{x_{i+1}}]$ equals $2 \cdot \Pr[A(x_1, ..., x_i) = (-1)^{x_{i+1}}] - 1$). The original proof extends in a straightforward manner; see Appendix A.3.

The following result presents a conditional transformation from next-bit unpredictability (by deterministic algorithms) to pseudorandomness (which holds with respect to probabilistic algorithms). This transformation relies on the fact that the hypothesis $\mathcal{BPP} = \mathcal{P}$ allows for derandomizing potential probabilistic next-bit predictors, which are obtained (in the usual way) from potential distinguishers.

Theorem A.3 (next-bit unpredictability implies pseudorandomness): *If $\mathcal{BPP} = \mathcal{P}$, then, for every polynomial p, there exists a polynomial p' such that the following holds: If G satisfies the first condition of Definition 4.3 and is next-bit p'-unpredictable, then G is a p-robust canonical derandomizer.*

[36] That is, the expected payoff with respect to a random variable $\zeta \in \{0,1\}$ is maintained, because the original payoff is $\mathrm{E}[(1-2p) \cdot (-1)^\zeta]$, whereas the payoff of the resulting Boolean predictor is $p \cdot \mathrm{E}[(-1)^{\zeta+1}] + (1-p) \cdot \mathrm{E}[(-1)^\zeta]$.

Indeed, an unconditioned (but weaker) version of Theorem A.3 (i.e., one that does not assume $\mathcal{BPP} = \mathcal{P}$ (but considers only quadratic-time deterministic distinguishers)) was discussed in [10], and was observed to imply some derandomization (albeit weaker than the one stated in Proposition 4.4, since [10] could not allow probabilistic distinguishers in Definition 4.3). Goldreich and Wigderson saw this implication as evidence to the unlikeliness of proving such a version [10]. Our point, however, is that the assumption $\mathcal{BPP} = \mathcal{P}$ does allow to prove that next-bit unpredictability implies pseudorandomness (in an adequate sense (i.e., as in Definition 4.3)).

Proof: Suppose towards the contradiction that G is not pseudorandom in the sense of Definition 4.3. That is, there exists a probabilistic p-time distinguisher with a distinguishing gap of $\delta(k)$ that for infinitely many k is larger than $1/p(\ell(k))$. Applying the standard transformation from distinguishing to predicting (cf., e.g., [8, Sec. 3.3.5]), we obtain a probabilistic p-time predictor A that outputs a binary value (say in $\{-1,1\}$) and has a correlation of at least $\epsilon(k) \stackrel{\text{def}}{=} \delta(k)/\ell(k)$ in guessing the next bit of G (wrt a prefix of some length).[37] More precisely, for infinitely many k, there exists $i < \ell(k)$, such that

$$\mathrm{E}[A(1^{\ell(k)}, x_1, ..., x_i) \cdot (-1)^{x_{i+1}}] \geq \epsilon(k), \tag{20}$$

where $(x_1, ..., x_{\ell(k)}) \leftarrow G(s)$ for a uniformly selected $s \in \{0,1\}^k$.

Next, we consider a probabilistic FPTAS for approximating the quantity q such that $q(1^{\ell(k)}, x) \stackrel{\text{def}}{=} \mathrm{E}[A(1^{\ell(k)}, x)]$. Note that $q : \{0,1\}^i \to [-1,1]$ has the same correlation that A has with the $i+1^{\text{st}}$ bit of G, because for every x and every random variable $\zeta \in \{0,1\}$ it holds that $\mathrm{E}[q(x) \cdot (-1)^\zeta] = \mathrm{E}[A(x) \cdot (-1)^\zeta]$. Thus, our aim is to obtain a deterministic FPTAS for q, which we do by noting that the existence a probabilistic FPTAS is straightforward, and invoking Corollary 3.6. Details follow.

A probabilistic FPTAS for q is obtained by invoking A for $O(\ell/\epsilon^2)$ times and outputting the average value.[38] This yields an algorithm A' with output in $[-1,+1]$ such that for every x it holds that $\Pr[A'(x) = q(x) \pm \epsilon/4] > 2/3$. At this point we invoke Corollary 3.6 (or rather its furthermore part), and obtain a deterministic algorithm A'' that satisfies $A'(x) = q(x) \pm \epsilon/2$ for every x. Furthermore, A'' has polynomial running time, where the polynomial p' only depends on the polynomial p (since p determines the running time of A as well as ϵ). Note that for every x and every random variable $\zeta \in \{0,1\}$ it holds that $\mathrm{E}[A''(x) \cdot (-1)^\zeta] > \mathrm{E}[q(x) \cdot (-1)^\zeta] - \epsilon/2$, which implies that A'' has correlation at least $\epsilon(k)/2$ with the next-bit of $G(U_k)$ for infinitely many k. Since the entire argument can be applied to any p-time distinguisher, yielding a p'-time predictor of correlation greater than $1/p'$ (for the same p', which only depends on p), we derive a contradiction to the p'-unpredictability hypothesis. The theorem follows. ■

Comment: The generator constructed in the foregoing proof does not satisfy the notion of a canonical derandomizer stated in Definition 4.2. Indeed, in this case, a probabilistic polynomial-time finder F, which runs for more time than the foregoing generator, can find a string x that allows very fast distinguishing. The details are as at the end of Appendix A.2.

A.3 Proof of Theorem A.2

It will be more convenient to restate Theorem A.2 it terms of $\{-1,1\}$. Thus, the generator outputs sequences over $\{-1,1\}$ rather than sequences over $\{0,1\}$. Also, it will be convenient to consider

[37] As stressed in [10], the resulting predictor is probabilistic even if we start with a deterministic distinguisher.

[38] Actually, this does not yield a FPTAS, but rather an approximation up to an additive term of $\epsilon/2$, We can present this as a FPTAS for the value of $q + 2 \in [1,3]$, and so proceed via the FPTAS formalism.

constructing the support of the generator, and assuming that it just outputs a strong that is uniformly distributed in its support. Since we are interested in generators of exponential stretch, these (support) sets have size that is polynomial in the length of the strings in them, since the seed length is logarithmic in these set sizes. Specifically, we refer to sets of the form $S = \cup_{n\in\mathbb{N}} S_n$, where $S_n \subset \{-1,1\}^n$, and say that such a set is polynomial-time constructible if there exists a polynomial-time algorithm that on input 1^n outputs the list of all sequences in S_n.

Theorem A.4 (Theorem A.2, restated): *For every polynomial p, there exists a polynomial-time constructible set $S = \cup_{n\in\mathbb{N}} S_n$ such that, for every deterministic algorithm A of running-time p and output in $[-1,+1]$, and for all sufficiently large n and all $i < n$, it holds that $\mathrm{E}[A(x_1,...,x_i)\cdot x_{i+1}] \leq 1/p(n)$, where $x = (x_1,...,x_n)$ is uniformly selected in S_n.*

The following text was essentially reproduced from [10], while introducing the necessary adaptations (which are quite minor).

Proof: Consider an enumeration of (modified)[39] deterministic machines running within time $p(n)$ on input of length n, and suppose that we wish to fool the $m = m(n) < p(n)$ first machines in the sense that we wish to upper bound its correlation with the next bit better than ϵ, where $\epsilon = 1/p(n)$. Let M be one of the machines we wish to fool.

We construct S_n in (roughly) n iterations, such that in iteration i we construct $S_{n,i} \subseteq \{-1,1\}^i$. We start with $S_{n,k} = \{-1,1\}^k$, where $k = 2\log_2(m/\epsilon)$, and let $K = 2^k$. In the $i+1^{\text{st}}$ iteration, we consider the vector $v_M \in [-1,+1]^K$ representing the output of M on each of the K possible i-bit long strings; that is, $v_M[j] = M(x^{(j)})$, where $x^{(j)}$ is the j^{th} string in $S_{n,i} \subseteq \{-1,1\}^i$. (This represents M's attempt to correlate its output with the $i+1^{\text{st}}$ bit of a string uniformly selected in S_n, based on the i-bit long prefix of that string.) Our aim is to find a vector $u \in \{-1,1\}^K$ that has low correlation with all the v_M's. Once such a vector u is found, we extend $S_{n,i}$ into $S_{n,i+1}$ in the natural manner; that is,

$$S_{n,i+1} \stackrel{\text{def}}{=} \{x^{(j)}u[j] : \text{ where } x^{(j)} \text{ is the } j^{\text{th}} \text{ string in } S_{n,i}\} \subset \{-1,1\}^{i+1}. \tag{21}$$

It follows that $S_n \stackrel{\text{def}}{=} S_{n,n}$ satisfies the small correlation requirement; that is, for each of the above M's and for each $i < n$, the correlation of $M(x_1\cdots x_i)$ with x_{i+1}, when x is uniformly selected in S_n, is at most ϵ.

It remains to specify how a suitable vector $u \in \{-1,1\}^K$ is found, in each iteration. This is done by using a pairwise independent sample space for strings of length K, while recalling that such spaces can be constructed in poly(K)-time (cf. [3]). Thus, it suffices to show that such a sample space must always contain a suitable vector $u \in \{-1,1\}^K$. This is quite an immediate consequence of the following claim.

Claim: Let $v \in [-1,+1]^K$ be arbitrary, and u be a sequence of K uniformly-distributed pairwise-independent elements of $\{-1,1\}$. Then, the probability that the correlation between u and v is greater than ϵ (i.e., $\sum_i u_i v_i > \epsilon K$) is strictly less than $\frac{1}{\epsilon^2 K}$.

Proof: For each $j \in [K]$, we define a random variable $\eta_j \in [-1,+1]$ such that $\eta_j \stackrel{\text{def}}{=} v[j]\cdot u[j]$. Since v is fixed and u is a sequence of K uniformly-distributed pairwise-independent bits, it follows that

[39] Recall that one cannot effectively enumerate all machines that run within some given time bound. Yet, one can enumerate all machines, and modify each machine in the enumeration such that the running-time of the modified machine respects the given time bound, while maintaining the functionality of the original machines in the case that the original machine respects the time bound. This is done by simply incorporating a time-out mechanism.

the η_j's are pairwise-independent and $\mathrm{E}(\eta_j) = 0$ for each $j \in [K]$. Using Chebyshev's Inequality, we have

$$\begin{aligned} \Pr\left[\left|\sum_{j\in[K]} \eta_j\right| \geq \epsilon \cdot K\right] &\leq \frac{\mathrm{Var}(\sum_j \eta_j)}{(\epsilon K)^2} \\ &= \frac{1}{\epsilon^2 K} \end{aligned}$$

and the claim follows. □

Since $\epsilon^2 K > m$, it follows that the aforementioned sample space contains a vector u that has correlation at most ϵ with each of the m vectors representing the m machines (i.e., the vectors v_M's). Thus, we construct the desired set S_n in time polynomial in n, and the theorem follows. ■