Derandomizing from Random Strings

Harry Buhrman[*]
CWI and University of Amsterdam, Amsterdam
buhrman@cwi.nl

Lance Fortnow[†]
Northwestern University
fortnow@northwestern.edu

Michal Koucký[‡]
Institute of Mathematics, AS CR
koucky@math.cas.cz

Bruno Loff[§]
CWI, Amsterdam
bruno.loff@gmail.com

September 16, 2010

Abstract

In this paper we show that BPP is truth-table reducible to the set of Kolmogorov random strings R_K. It was previously known that PSPACE, and hence BPP is Turing-reducible to R_K. The earlier proof relied on the adaptivity of the Turing-reduction to find a Kolmogorov-random string of polynomial length using the set R_K as oracle. Our new non-adaptive result relies on a new fundamental fact about the set R_K, namely each initial segment of the characteristic sequence of R_K has high Kolmogorov complexity. As a partial converse to our claim we show that strings of very high Kolmogorov-complexity when used as advice are not much more useful than randomly chosen strings.

1 Introduction

Kolmogorov complexity studies the amount of randomness in a string by looking at the smallest program that can generate it. The most random strings are those we cannot compress at all. What if we could know which strings are Kolmogorov-random and which are not, by making queries to the oracle $R_K = \{x \mid K(x) \geq |x|\}$?

[*]Partially supported by an NWO VICI grant.
[†]Supported in part by NSF grants CCF-0829754 and DMS-0652521.
[‡]Partially supported by GA ČR 201/07/P276 and P202/10/0854, project No. 1M0021620808 of MŠMT ČR, Institutional Research Plan No. AV0Z10190503 and grant IAA100190902 of GA AV ČR.
[§]Supported by FCT grant no. SFRH/BD/43169/2008.

Allender et al. [**?**] showed that this gives us surprising computational power. It was shown, for instance, that polynomial time adaptive (Turing) access to R_K enables one to do PSPACE-computations, i.e., PSPACE $\subseteq P^{R_K}$. One of the ingredients in the proof was to show how one may, on input 0^n and in polynomial time with *adaptive* access to R_K, generate a polynomially long Kolmogorov random string. However, a polynomial-time algorithm with *non-adaptive* access to R_K, on input 0^n, can only generate random strings of length at most $O(\log n)$.

In an attempt to *characterize* PSPACE as the class of sets reducible to R_K, Allender, Buhrman and Koucký [**?**] noticed that this question depends on the choice of universal machine used in the definition of the notion of Kolmogorov complexity. They also started a systematic study of weaker and non-adaptive access to R_K. They showed for example that

$$\mathrm{P} = \mathrm{REC} \cap \bigcap_U \{A \mid A \leq^{\mathrm{p}}_{\mathrm{dtt}} R_{K_U}\}.$$

This result and the fact that with non-adaptive access to R_K in general only logarithmically small strings can be found seems to suggest that adaptive access to R_K is needed in order to be useful.

Our first result proves this intuition false: We show that polynomial time non-adaptive access to R_K can be used to derandomize any BPP computation. In order to derandomize a BPP computation one needs a (pseudo)random string of polynomial size. As mentioned before one can only obtain short, $O(\log n)$ sized, random strings from R_K. Instead we show that the characteristic sequence formed by the strings of length $c \log n$, $R_K^{=c \log n}$, itself a strings of length n^c, is complex enough to figure as a hard function in the hardness versus randomness framework of Impagliazzo and Wigderson [**?**]. This way we construct a pseudorandom generator that is strong enough to derandomize BPP.

In particular we show that for every time bound t, there is a constant c such that $R_K \notin$ i.o.-DTIME$(t)/2^{n-c}$. This is in stark contrast with the time-unbounded case where only n bits of advice are necessary [**?**]. As a consequence we give an alternative proof of the existence of an r.e. set A, due to Barzdin [**?**], such that for all time bounds t, there exists c_t such that $K^{t(n)}(A_{1:n} \mid n) \geq n/c_t$. We simply take for A the complement of R_K. We also show that infinitely-often is required for this tight bound.

Next we try to establish whether we can characterize BPP as the class of sets that non-adaptively reduce to R_K. One can view the truth-table reduction to R_K as a computation with advice of $K^{t(n)}$ complexity $\Omega(n)$. We can show that for sets in EXP and $t(n) \in 2^{n^{\Omega(1)}}$, polynomial-time computation with polynomial (exponential, resp.) size advice of $K^{t(n)}$ complexity $n - O(\log n)$ $(n - O(\log\log n)$, resp.) can be simulated by bounded error probabilistic machine with almost linear size advice. For paddable sets that are complete for NP, $\mathrm{P}^{\#\mathrm{P}}$, PSPACE, or EXP we do not even need the linear size advice. Hence, advice of high $K^{t(n)}$ complexity is no better than a truly random string.

Summarizing our results:

- For every computable time bound t there is a constant c (depending on t) such that $R_K \notin$ i.o.-DTIME$(t)/2^{n-c}$.
- The complement of R_K is a natural example of an computably enumerable set whose characteristic sequence has high time bounded Kolmorogov complexity for every n.
- BPP is truth-table reducible to R_K.
- A poly- up-to exponential-size advice that has very large $K^{t(n)}$ complexity can be replaced by $O(n \log n)$ bit advice and true randomness.
- For every c, $R_K \in$ i.o.-REC$/2^{n-c}$.
- For some c, $R_K \notin$ REC$/2^{n-c}$.

2 Preliminaries

We remind the reader of some of the definitions we use. Let M be a Turing machine. For any string $x \in \{0,1\}^*$, the Kolmogorov complexity of x relative to M is $K_M(x) = \min\{\ |p| \mid p \in \{0,1\}^* \ \&\ M(p) = x\}$, where $|p|$ denotes the length of string p. It is well known that for a *universal* Turing machine U and any other machine M there is a constant c_M such that for all strings x, $K_U(x) \leq K_M(x) + c_M$. For the rest of the paper we will fix some universal Turing machine U and we will measure Kolmogrov complexity relative to that U. Thus, we will not write the subscript U explicitly.

For a time bound t we define $K^t_U(x) = \min\{\ |p| \mid U(p) = x$ and $U(p)$ uses at most $t(|x|)$ steps$\}$. Unlike traditional computational complexity the time bound is a function of the length of the output of U. Similarly to the generic Kolmogorov complexity there is an *efficient* universal Turing machine U such that for any other machine M and time bounds t and t', where $t \in \omega(t' \log t')$, there exists a constant c_M such that for all strings x, $K^t_U(x) \leq K^{t'}_M(x) + c_M$ and $K_U(x) \leq K_M(x) + c_M$. For the rest of the paper we fix such a machine U and measure time-bounded Kolmogorov complexity relative to it.

A string x is said to be *Kolmogorov-random* if $K(x) \geq |x|$. The set of Kolmogorov-random strings is denoted by $R_K = \{x \in \{0,1\}^* \mid K(x) \geq |x|\}$. For an integer n and set $A \subseteq \{0,1\}^*$, $A^{=n} = A \cap \{0,1\}^n$. The following well known claim can be proven by showing a lower bound on the Kolmogorov complexity of $|R^{=n}_K|$ (see [**?**]).

Proposition 1. *There is a constant $d > 0$ such that for all n, $|R^{=n}_K| \geq 2^n/d$.*

Similar notation is also used for the time-bounded Kolmogorov complexity for which the previous proposition is also valid.

We also use computation with advice. We deviate slightly from the usual definition of computation with advice in the way how we express and measure the running time. For an *advice function* $\alpha : \mathbf{N} \to \{0,1\}^*$, we say that $L \in \mathrm{P}/\alpha$ if there is a Turing machine

M such that for every $x \in \{0,1\}^*$, $M(x, \alpha(|x|))$ runs in time polynomial in the length of x and $M(x, \alpha(|x|))$ accepts iff $x \in L$. We assume that M has random access to its input so the length of $\alpha(n)$ can grow faster than any polynomial in n. Similarly, we define EXP$/\alpha$ where we allow the machine M to run in exponential time in length of x on the input $(x, \alpha(|x|))$. Furthermore, we are interested not only in Boolean languages (decision problems) but also in functions, so we naturally extend both definitions also to computation with advice of functions. Typically we are interested in the amount of advice that we need for inputs of length n so for $f : \mathbf{N} \to \mathbf{N}$, C/f is the union of all C/α for α satisfying $|\alpha(n)| \leq f(n)$.

Let L be a language and C be a language class. We say that $L \in$ i.o.$-C$ if there exists a language $L' \in C$ such that for infinitely many n, $L^{=n} = L'^{=n}$. For a Turing machine M, we say $L \in$ i.o.-M$/f$ if there is some advice function α with $|\alpha(n)| \leq f(n)$ such that for infinitely many n, $L^{=n} = \{x \in \Sigma^n \mid M(x, \alpha(|x|)) \text{ accepts}\}$. The definitions are similar for functions instead of languages.

We will refer to probabilistic computation with advice. For $f : \mathbf{N} \to \mathbf{N}$, $BPP//f$ is the class of sets decidable by a bounded error probabilistic Turing machine with an advice function α such that $|\alpha(n)| \leq f(n)$, where the machine is only required to have a bounded error when given the correct advice.

We say that a set A polynomial-time Turing reduces to a set B, if there is an oracle machine M that on input x runs in polynomial time and with oracle B decides whether $x \in A$. If M asks its questions *non-adaptively*, i.e., each oracle question does not depend on the answers to the previous oracle questions, we say that A polynomial-time truth-table reduces to B ($A \leq^{\mathrm{p}}_{\mathrm{tt}} B$). Moreover, $A \leq^{\mathrm{p}}_{\mathrm{dtt}} B$ if machine M outputs as its answer the disjunction of the oracle answers. Similarly, $A \leq^{\mathrm{p}}_{\mathrm{ctt}} B$ for the conjunction of the answers.

3 High circuit complexity of R_K

In this section we prove that the characteristic sequence of R_K has high circuit complexity almost everywhere. We will first prove the following lemma.

Lemma 2. *For every total Turing machine M there is a constant c_M such that R_K is not in* i.o.-M$/2^{n-c_M}$.

There is a (non-total) Turing machine M such that R_K is in $M/n+1$ where the advice is the number of strings in $R_K^{=n}$. With this advice, M can find all the non-random strings of length n, but will not halt if the advice underestimates the number of random strings.

Proof of Lemma 2. Suppose we have a total Turing machine M, and that, for some advice α of length k, $x \in R_K^{=n} \iff (x, \alpha) \in L(M)$ for infinitely many lengths n.

For strings β of length k, let $R_\beta = \{x \in \Sigma^n \mid (x, \beta) \in L(M)\}$. By Proposition 1, for some integer $d > 0$, $|R_\alpha| \geq 2^n/d$. So we know that if $|R_\beta| < 2^n/d$ then $\beta \neq \alpha$. We call β *good* if $|R_\beta| \geq 2^n/d$.

Fix a good β and choose $x_1, \ldots, x_m$ at random. The probability that all the x_i are not in R_β is at most $(1 - 1/d)^m < 2^{-m/d}$. There are 2^k advice strings β of length k, and so if we take $m = dk$, then $2^{-m/d} \leq 2^{-k}$. In that case we can find a sequence $x_1, \ldots, x_m$ such that every good β will have an $x_i \in R_\beta$ for some i.

We may computably search all such sequences, so let $x_1, \ldots, x_m$ be the lexicographically least sequence such that each good β has some $x_i \in R_\beta$. Each x_i can be described by giving n, d, i and programs for M and the search procedure. Let $a = n - \lceil \log m \rceil$ with a self-delimiting description of length at most $a/2 + 4$. Then for a constant s depending only on M, each x_i can be described by $\lceil \log m \rceil + a/2 + 4 + 2 \log d + s \leq \log k + \log d + \frac{1}{2}(n - \log k - \log d) + 6 + 2 \log d + s \leq \frac{1}{2}(n + \log k) + 3 \log d + 6 + s$ bits. Set $c_M = 6 \log d + 12 + 2s$.

Now we must have $k \geq 2^{n - c_M}$ as otherwise we obtain a contradiction with the fact that at least one of the x_i's must be random since it is from R_α. □

In order to get our statement about time bounded advice classes we instantiate Lemma 2 with universal machines U_t that run for t steps, use the first part of their advice, in prefix free form, as a code for a machine to simulate and have the second part of the advice for U_t as the advice for the simulated machine. The following is a direct consequence of Lemma 2.

Lemma 3. *For every computable time bound t and universal advice machine U_t there is a constant c_t such that R_K is not in* $\text{i.o.-U}_t/2^{n-c_t}$.

We are now ready to prove the main theorem from this section.

Theorem 4. *For every computable time bound t there is a constant d_t such that R_K is not in* $\text{i.o.-DTIME}(t)/2^{n-d_t}$.

Proof. Suppose the theorem is false, that is there is a time bound t such that for every d there is a machine M_d that runs in time t such that $R_k \in \text{i.o.-M}_d/2^{n-d}$. Set $t' = t \log^2 t$ and let $c_{t'}$ be the constant that comes out of Lemma 3 when instantiated with time bound t'. Set $d = c_{t'} + 1$ and let the code of machine M_d from the (false) assumption have size e. So we have that $R_k \in \text{i.o.-M}_d/2^{n-d}$. This in turn implies that $R_K \in \text{i.o.-U}_{t'}/2^{n-d} + e + 2 \log e$, which implies that $R_K \in \text{i.o.-U}_{t'}/2^{n-c_{t'}}$ a contradiction with Lemma 3. The last step is true because the universal machine running for at most $t' = t \log^2 t$ steps, can simulate M_d, which runs in time t.

□

As an immediate corollary we get an alternative, more natural candidate for Barzdin's computably enumerable set that has high resource bounded Kolomorov complexity, namely the set of compressible strings.

Corollary 5. *For every computable time bound t there is a constant c such that $K^t(\overline{R_k}(1:n) \mid n) \geq n/c$*

Barzdin [**?**] also showed that this lower bound is optimal. That is the dependence of c on the time bound t is needed for the characteristic sequence of every r.e. set. Hence the dependence on t is also necessary in our Theorem 4. Indeed it can be shown that:

Theorem 6. *(a) For every c, there is a time bound t with $R_K \in \text{i.o.-DTIME}(t)/2^{n-c}$.*

(b) There exists some c such that, for every time bound t, $R_K \notin \text{DTIME}(t)/2^{n-c}$.

Theorem 6 gives us an interesting contrast.

Corollary 7. *The following hold:*

(a) For every c, $R_K \in \text{i.o.-REC}/2^{n-c}$.

(b) For some c, $R_K \notin \text{REC}/2^{n-c}$. The constant c depends only on the universal machine defining R_K.

Proof of Theorem 6. (a) Let H denote the binary entropy function, and choose $0 < \alpha < 1/2$, such that $H(\alpha) < 2^{-c}$. Let $\sigma = \liminf_{n\to\infty} |\overline{R_K^{=n}}|/2^n$, and set $\tilde{\sigma}$ to the dyadic rational written with the first $1 + \log(1/\alpha)$ bits of σ. Note $\tilde{\sigma}$ is independent of n and can be hardwired into our machine.

Then, on input 0^n, we can (1) enumerate $\tilde{\sigma}2^n$ many non-random strings of length n, and (2) assume that some given advice tells us, among the strings which we did not enumerate, which are in R_K. Since for sufficiently large n, we can always find a fraction of $\tilde{\sigma}$ non-random strings of length n, this algorithm halts almost everywhere. And because, for infinitely many n, $\left|\tilde{\sigma}2^n - |\overline{R_K^{=n}}|\right| < \alpha 2^n$, after (1) we are left with at most $\alpha 2^n$ non-random strings still to be enumerated. Since $\log \sum_{k\leq\alpha 2^n} \binom{2^n}{k} \leq H(\alpha)2^n \leq 2^{n-c}$ (cf. [**?**, p.283]), then 2^{n-c} many bits of advice will be able to point out exactly these strings.

(b) Fix d such that $|R_K^{=n}| > n/d$ for all n.

Let $M_1, M_2, M_3, \ldots$ enumerate all Turing machines, and let ℓ be the length of a program which:

1. given an input i of size $n - \ell - 1$,
2. will work with $M = M_n$, and, as in the proof of Lemma 2, assuming M is total,
3. will find the lexicographically least sequence of n-bit strings $x_1, \ldots, x_m$, with $m = 2^{n-\ell-1}$, such that for any good advice β of length 2^{n-c}, with $c = \log d + \ell + 1$, there is some x_j such that $M(x_j, \beta) = 1$; and finally
4. outputs x_i.

The values in the program are allowed to depend on ℓ by the use of Kleene's fixed point theorem. Then notice that if M_n is total, regardless of which advice β, of length 2^{n-c}, it is given, there will always be some n-bit string x_j which is not random (it is described using $\ell + \log m < n$ bits), and for which $M_n(x_j, \beta) = 1$. So M_n does not decide R_K. □

We can also obtain a variant of Theorem 4 for the time-bounded Kolmogorov complexity.

Lemma 8. *There exists a constant c such that for any time bounds t and t', if $t \in \omega(2^{2^{n+c}} t'(n) \log t'(n))$ then for every Turing machine M running in time t' there is a constant c_M such that R_{K^t} is not in* i.o.-M$/2^{n-c_M}$.

Proof. An inspection of the proof of Lemma 2 reveals that the running time of the search procedure for $x_1, \ldots, x_m$ can be bounded by $2^{2^{n+c'}} t'(n)$ for some universal constant c' independent of M. Thus by the same argument as in that proof using the same notation and the properties of time-bounded Kolmogorov complexity, for each x_i we have $K^t(x_i) \le \frac{1}{2}(n + \log k) + 3 \log d + 6 + s$ bits. This would lead to a contradiction unless $k \ge 2^{n-c_M}$. □

Using the same arguments as before we obtain the following corollary.

Corollary 9. *There exists a constant c such that for any time bounds t and t', if $t \in \omega(2^{2^{n+c}} t'(n) \log^3 t'(n))$ then there is a constant $d_{t'}$ such that R_{K^t} is not in* i.o.-DTIME$(t')/2^{n-d_{t'}}$.

For example, if we instantiate $t = 2^{2^{2n}}$ and $t' = 2^{n^{\log n}}$ in the above corollary we get that R_{K^t} is not in i.o.-EXP$/2^{n-d_{t'}}$.

4 BPP truth-table reduces to R_k

In this section we investigate what languages are reducible to R_k. We start with the following theorem which one can prove using nowadays standard derandomization techniques.

Theorem 10. *Let $\alpha : \{0\}^* \to \{0,1\}^*$ be a length-preserving function and $\delta > 0$ be a constant, such that $\alpha(0^n) \notin$ i.o.-P$/n^\delta$. Then for every $A \in$ BPP there exists $d > 0$ such that $A \in$ P$/\alpha(0^{n^d})$.*

Proof. $\alpha(0^n) \notin$ i.o.-P$/n^\delta$ implies that when $\alpha(0^n)$ is interpreted as a truth-table of a function $f_{\alpha(0^n)} : \{0,1\}^{\log n} \to \{0,1\}$, $f_{\alpha(0^n)}$ does not have boolean circuits of size $n^{\delta/3}$ for all n large enough. It is known that such a function can be used to build the Impagliazzo-Wigderson pseudorandom generator [?] which can be used to derandomize boolean circuits of size $n^{\delta'}$ for some $\delta' > 0$ (see [?, ?, ?]). Hence, bounded-error probabilistic computation running in time n^ℓ can be derandomized in polynomial time given access to $\alpha(0^{n^{2\ell/\delta'}})$. □

From Theorem 4 and the above Theorem we obtain the following corollary.

Corollary 11. BPP $\leq_{tt}^{p} R_K$.

Proof. Let $\alpha(0^n)$ be the truth-table of R_K on strings of length $\lfloor \log n \rfloor$ padded by zeros to the length of n. By Theorem 4, $\alpha(0^n) \notin$ i.o.-P$/(n/c)$ for some $c > 0$. Consider any $A \in$ BPP. By Theorem 10 for some d, $A \in P/\alpha(0^{n^d})$. The claim follows by noting that a truth-table reduction to R_k may query the membership of all the strings of length $\lfloor \log n^d \rfloor$ to construct $\alpha(0^{n^d})$ and then run the $P/\alpha(0^{n^d})$ algorithm for A. □

A similar argument establishes a variant of the claim for time-bounded Kolmogorov complexity.

Corollary 12. *For any time bound* $t \in \Omega(2^{2^{2n}})$*,* BPP $\leq_{tt}^{p} R_{K^t}$.

It is interesting to compare this statement with the result of Buhrman and Mayordomo [?] that EXP $\not\subseteq P^{R_{K^t}}$ for $t = 2^{n^2}$. Hence, it will be rather difficult to prove the above corollary for $t = 2^{n^2}$.

Our goal would be to show that using R_K as a source of randomness is the only way to make use of it. Ideally we would like to show that any recursive set that is truth-table reducible to R_K must be in BPP. We fall short of such a goal. However we can show the following claim.

Theorem 13. *Let* $\alpha : \{0\}^* \to \{0,1\}^*$ *be a length-preserving function and* $c > 0$ *be a constant, such that* $\alpha(0^n) \notin$ i.o.-EXP$/n - c \log n$*. Then for every* $A \in$ EXP *if* $A \in$ P$/\alpha(0^{n^d})$ *for some* $d > 0$ *then* $A \in$ BPP$//O(n \log n)$*.*

This theorem says that very Kolmogorov-random advice of polynomial size can be replaced by almost linear size advice and true randomness. It can be proven using diagonalization that such advice functions exist, and these can be used to derandomize BPP as we did above.

We come short of proving a converse of Corollary 11 in two respects. First, the advice is supposed to model the initial segment of the characteristic sequence of R_K which the truth-table can access. However, by providing only polynomial size advice we restrict the hypothetical truth-table reduction to query strings of only logarithmic length. Second, the randomness that we require from the initial segment is much stronger than what one can prove and what is in fact true for the initial segment of the characteristic sequence of R_K. One can deal with the first issue as is shown by Theorem 14 but we do not know how to deal with the second one.

Proof. Let M be a polynomial time Turing machine and $A \in$ EXP be a set such that $A(x) = M(x, \alpha(|x|^d))$. We claim that for all n large enough there is a non-negligible fraction of advice strings r of size n^d that could be used in place of $\alpha(n^d)$ more precisely:

$$\Pr_{r \in \{0,1\}^{n^d}} [\forall x, x \in A \iff M(x,r) = 1] > \frac{1}{n^{cd}}.$$

To prove the claim consider the set $G = \{r \in \{0,1\}^{n^d};\ \forall x \in \{0,1\}^n, x \in A \iff M(x,r) = 1\}$. Clearly, $G \in \mathrm{EXP}$ and $\alpha(0^{n^d}) \in G$. If $|G^{=n^d}| \le 2^{n^d}/n^{cd}$ then $\alpha(0^{n^d})$ can be computed in exponential time from its index in the set $G^{=n^d}$ of length $n^d - cd\log n$. Since $\alpha(0^{n^d}) \notin$ i.o.-EXP$/n^d - cd\log n$ this cannot happen infinitely often.

Now we present an algorithm which on input x, and using only $O(n\log n)$ bits of advice (in fact $O(\log n)$ entries from the truth table of A), will with high probability produce a string in $r \in G$, and output $A(x)$:

1. Given an input x of length n, and an advice string $x_1, A(x_1), ..., x_k, A(x_k)$,
2. sample at most $2n^{cd}$ strings of length n^d until the first string r is found such that $M(x_i, r) = A(x_i)$ for all $i \in \{1, \ldots, k\}$.
3. If we find r consistent with the advice then output $M(x,r)$, otherwise output 0.

For all n large enough the probability that the second step does not find r compatible with the advice is upper-bounded by the probability that we do not sample any string from G which is at most $(1 - \frac{1}{n^{cd}})^{2n^{cd}} < e^{-2} < 1/6$.

It suffices to show that we can find an advice sequence such that, for at least 4/5-fraction of the r's compatible with the advice, $M(x,r) = A(x)$. For given n, we will find the advice by pruning iteratively the set of bad random strings $B = \{0,1\}^{n^d} \setminus G$. Let $i = 0, 1, \ldots, \log_{5/4} 4n^{cd}$. Set $B_0 = B$. If there is a string $x \in \{0,1\}^n$ such that for at least 1/5 of $r \in B_i$, $M(x,r) \ne A(x)$, then set $x_{i+1} = x$ and $B_{i+1} = B_i \cap \{r \in \{0,1\}^{n^d} \mid M(x_{i+1}, r) = A(x_{i+1})\}$. If there is no such string x then stop and the x_i's obtained so far will form our advice. Notice, if we stop for some $i < \log_{5/4} 4n^{cd}$ then for all $x \in \{0,1\}^n$, $\Pr_{r \in \mathcal{B}_i}[M(x,r) \ne A(x)] < 1/5$. Hence, for any given input, the r found by the algorithm to be compatible with the advice will give the correct answer with probability at least 4/5. On the other hand, if we stop building the advice at $k = \log_{5/4} 4n^{cd}$ then $|B_k| \le 2^{n^d} \cdot (4/5)^{\log_{5/4} 4n^{cd}} \le |G^{=n^d}|/4$. Hence, any string r found by the algorithm to be compatible with the advice $x_1, A(x_1), ..., x_k, A(x_k)$ will come from G with probability at least 4/5. □

The following theorem can be established by a similar argument. It again relies on the fact that a polynomially large fraction of all advice strings of length 2^{n^d} must work well as an advice. By a pruning procedure similar to the proof of Theorem 13 we can avoid bad advice. In the BPP algorithm one does not have to explicitly guess the whole advice but only the part relevant to the pruning advice and to the current input.

Theorem 14. *Let $\alpha : \{0\}^* \to \{0,1\}^*$ be a length preserving function and $c > 0$ be a constant. If $\alpha(0^n) \notin$ i.o.-EXP$/n - c\log\log n$ then for every $A \in$ EXP if $A \in \mathrm{P}/\alpha(0^{2^{n^d}})$ for some $d > 0$ then $A \in \mathrm{BPP}//O(n\log n)$.*

We show next that if the set A has some suitable properties we can dispense with the linear advice all together and replace it with only random bits. Thus for example if $\mathrm{SAT} \in \mathrm{P}/\alpha(0^n)$ for some computationally hard advice $\alpha(0^n)$ then $\mathrm{SAT} \in \mathrm{BPP}$.

Theorem 15. *Let $\alpha : \{0\}^* \to \{0,1\}^*$ be a length preserving function and $c > 0$ be a constant such that $\alpha(0^n) \notin \text{i.o.-EXP}/n - c\log n$. Let A be paddable and polynomial-time many-one-complete for a class $\mathcal{C} \in \{\mathrm{NP}, \mathrm{P}^{\#\mathrm{P}}, \mathrm{PSPACE}, \mathrm{EXP}\}$. If $A \in \mathrm{P}/\alpha(0^{n^d})$ for some $d > 0$ then $A \in \mathrm{BPP}$ (and hence $C \subseteq \mathrm{BPP}$).*

To prove the theorem we will need the notion of instance checkers. We use the definition of Trevisan and Vadhan [**?**].

Definition 16. *An* instance checker *C for a boolean function f is a polynomial-time probabilistic oracle machine whose output is in $\{0, 1, \mathrm{fail}\}$ such that*

- *for all inputs x, $\Pr[C^f(x) = f(x)] = 1$, and*
- *for all inputs x, and all oracles f', $\Pr[C^{f'}(x) \notin \{f(x), \mathrm{fail}\}] \leq 1/4$.*

It is immediate that by linearly many repetitions and taking the majority answer one can reduce the error of an instance checker to 2^{-n}. Vadhan and Trevisan also state the following claim:

Theorem 17 ([**?**],[**?**, **?**])**.** *Every problem that is complete for* EXP, PSPACE *or* $\mathrm{P}^{\#\mathrm{P}}$ *has an instance checker. Moreover, there are* EXP*-complete problems,* PSPACE*-complete problems, and* $\mathrm{P}^{\#\mathrm{P}}$*-complete problems for which the instance checker C only makes oracle queries of length exactly $\ell(n)$ on inputs of length n for some polynomial $\ell(n)$.*

However, it is not known whether NP has instance checkers.

Proof. Proof of Theorem 15 To prove the claim for $\mathrm{P}^{\#\mathrm{P}}$-, PSPACE- and EXP-complete problems we use the instance checkers. We use the same notation as in the proof of Theorem 13, i.e., M is a Turing machine such that $A(x) = M(x, \alpha(|x|^d))$ and the set of good advice is G. We know from the previous proof that $|G^{=n^d}| \geq 2^{n^d}/n^{cd}$ because $\alpha(0^n) \notin \text{i.o.-EXP}/n - c\log n$.

Let C be the instance checker for A which on input of length n asks oracle queries of length only $\ell(n)$ and makes error on a wrong oracle at most 2^{-n}. The following algorithm is a bounded error polynomial time algorithm for A:

1. On input x of length n, repeat $2n^{cd}$ times
   (a) Pick a random string r of length $(\ell(n))^d$.
   (b) Run the instance checker C on input x and answer each of his oracle queries y by $M(y, r)$.

   (c) If C outputs fail continue with another iteration otherwise output the output of C.

2. Output 0.

Clearly, if we sample $r \in G$ then the instance checker will provide a correct answer and we stop. The algorithm can produce a wrong answer either if the instance checker always fails (so we never sample $r \in G$ during the iterations) or if the instance checker gives a wrong answer. Probability of not sampling good r is at most $1/6$. The probability of getting a wrong answer from the instance checker in any of the iterations is at most $2n^{cd}/2^n$. Thus the algorithm provides the correct answer with probability at least $2/3$.

To prove the claim for NP-complete languages we show it for the canonical example of SAT. The following algorithm solves SAT correctly with probability at least $5/6$:

1. On input ϕ of length n, repeat $2n^{cd}$ times
   (a) Pick a random string r of length n^d.
   (b) If $M(\phi, r) = 1$ then use the self-reducibility of SAT to find a presumably satisfying assignment a of ϕ while asking queries ψ of size n and answering them according to $M(\psi, r)$. If the assignment a indeed satisfies ϕ then output 1 otherwise continue with another iteration.
2. Output 0.

Clearly, if ϕ is satisfiable we will answer 1 with probability at least $5/6$. If ϕ is not satisfiable we will always answer 0. □

5 Open Problems

We have shown that the set R_K cannot be compressed using a computable algorithm and used this fact to reduce BPP non-adaptively to R_K. We conjecture that every computable set that non-adaptively reduces in polynomial-time to R_K sits in BPP and have shown a number of partial results in that direction.

The classification of languages that polynomial-time adaptively reduce to R_K also remains open. Can we characterize PSPACE this way?