SIAM J. COMPUT.
Vol. 30, No. 5, pp. 1485–1501

RANDOMNESS IS HARD*

HARRY BUHRMAN† AND LEEN TORENVLIET‡

Abstract. We study the set of incompressible strings for various resource bounded versions of Kolmogorov complexity. The resource bounded versions of Kolmogorov complexity we study are polynomial time *CD* complexity defined by Sipser, the nondeterministic variant *CND* due to Buhrman and Fortnow, and the polynomial space bounded Kolmogorov complexity *CS* introduced by Hartmanis. For all of these measures we define the set of random strings R_t^{CD}, R_t^{CND}, and R_t^{CS} as the set of strings x such that $CD^t(x)$, $CND^t(x)$, and $CS^s(x)$ is greater than or equal to the length of x for s and t polynomials. We show the following:

- $\mathrm{MA} \subseteq \mathrm{NP}^{\mathrm{R}_t^{CD}}$, where MA is the class of Merlin–Arthur games defined by Babai.
- $\mathrm{AM} \subseteq \mathrm{NP}^{\mathrm{R}_t^{CND}}$, where AM is the class of Arthur–Merlin games.
- $\mathrm{PSPACE} \subseteq \mathrm{NP}^{\mathrm{cR}_s^{CS}}$.

In the last item cR_s^{CS} is the set of pairs $\langle x, y\rangle$ so that x is random given y. These results show that the set of random strings for various resource bounds is hard for complexity classes under *nondeterministic* reductions.

This paper contrasts the earlier work of Buhrman and Mayordomo where they show that for polynomial time *deterministic* reductions the set of exponential time Kolmogorov random strings is not complete for EXP.

Key words. complexity classes, *CD* complexity, *CND* complexity, interactive proofs, Kolmogorov complexity, Merlin–Arthur, Arthur–Merlin, randomness, relativization

AMS subject classifications. 03D15, 68Q10, 68Q15, 68Q17, 68Q19

PII. S0097539799360148

1. Introduction.

The holy grail of complexity theory is the separation of complexity classes like P, NP, and PSPACE. It is well known that all of these classes possess complete sets and that it is thus sufficient for a separation to show that a complete set of one class is not contained in the other. Therefore lots of effort was put into the study of complete sets. (See [11].)

Kolmogorov [19], however, suggested focusing attention on sets which are *not* complete. His intuition was that complete sets possess a lot of "structure" that hinders a possible lower bound proof. He suggested to look at the set of time bounded Kolmogorov random strings. In this paper we will continue this line of research and study variants of this set.

Kolmogorov complexity measures the "amount" of regularity in a string. Informally the Kolmogorov complexity of a string x, denoted as $C(x)$, is the size of the smallest program that prints x and then stops. For any string x, $C(x)$ is less than or equal to the length of x (up to some additive constant). Those strings for which it holds that $C(x)$ is greater than or equal to the length of x are called *incompressible* or *random*. A simple counting argument shows that random strings exist.

*Received by the editors August 17, 1999; accepted for publication (in revised form) February 8, 2000; published electronically November 8, 2000.
http://www.siam.org/journals/sicomp/30-5/36014.html

†CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands (buhrman@cwi.nl). The work of this author was partially supported by the Dutch foundation for scientific research (NWO) by SION project 612-34-002, the European Union through NeuroCOLT ESPRIT Working Group 8556, HC&M grant ERB4050PL93-0516, and the EU fifth framework program QAIP, IST-1999-11234.

‡Department of Computer Science, University of Amsterdam, 24 Plantage Muidergracht, 1018 TV Amsterdam, The Netherlands (leen@wins.uva.nl).

In the sixties, when the theory of Kolmogorov complexity was developed, Martin [23] showed that the coRE set of Kolmogorov random strings is complete with respect to (resource unbounded) Turing reductions. Kummer [18] has shown that this can be strengthened to show that this set is also truth-table complete.

The resource bounded version of the set of random strings was first studied by Ko [17]. The *polynomial* time bounded Kolmogorov complexity $C^p(x)$ for p a polynomial is the smallest program that prints x in $p(|x|)$ steps [14]. Ko showed that there exists an oracle such that the set of random strings with respect to this time bounded Kolmogorov complexity is complete for coNP under strong nondeterministic polynomial time reductions. He also constructed an oracle where this set is not complete for coNP under deterministic polynomial time Turing reductions.

Buhrman and Mayordomo [10] considered the *exponential* time Kolmogorov random strings. The exponential time Kolmogorov complexity $C^t(x)$ is the smallest program that prints x in $t(|x|)$ steps for functions $t(n) = 2^{n^k}$. They showed that the set of $t(n)$ random strings is *not* deterministic polynomial time Turing hard for EXP. They showed that the class of sets that reduce to this set has p measure 0 and hence that this set is not even weakly hard for EXP.

The results in this paper contrast those from Buhrman and Mayordomo. We show that the set of random strings is hard for various complexity classes under *nondeterministic* polynomial time reductions.

We consider three well-studied measures of Kolmogorov complexity that lie in between $C^p(x)$ and $C^t(x)$ for p a polynomial and $t(n) = 2^{n^k}$. We consider the distinguishing complexity as introduced by Sipser [25]. The distinguishing complexity, $CD^t(x)$, is the size of the smallest program that runs in time $t(n)$ and accepts x and nothing else. We show that the set of random strings $\mathrm{R}_t^{CD} = \{x \mid CD^t(x) \geq |x|\}$ for t a fixed polynomial is hard for MA under nondeterministic reductions. MA is the class of Merlin–Arthur games introduced by Babai [1]. As an immediate consequence we obtain that BPP and $\mathrm{NP}^{\mathrm{BPP}}$ are in $\mathrm{NP}^{\mathrm{R}_t^{CD}}$.

Next we shift our attention to nondeterministic distinguishing complexity [6], $CND^t(x)$, which is defined as the size of the smallest *nondeterministic* algorithm that runs in time $t(n)$ and accepts only x. We define $\mathrm{R}_t^{CND} = \{x : CND^t(x) \geq |x|\}$ for t a fixed polynomial. We show that $\mathrm{AM} \subseteq \mathrm{NP}^{\mathrm{R}_t^{CND}}$, where AM is the class of Arthur–Merlin games [1]. It follows that the complement of the graph isomorphism problem, $\overline{GI}$, is in $\mathrm{NP}^{\mathrm{R}_t^{CND}}$ and that if for some polynomial t, $\mathrm{R}_t^{CND} \in \mathrm{NP} \cap \mathrm{coNP}$, then $GI \in \mathrm{NP} \cap \mathrm{coNP}$.

The $s(n)$ *space* bounded Kolmogorov complexity $CS^s(x|y)$ is defined as the size of the smallest program that prints x, given y, and uses at most $s(|x| + |y|)$ tape cells [14]. Likewise we define $\mathrm{cR}_s^{CS} = \{\langle x, y\rangle : CS^s(x|y) \geq |x|\}$ for $s(n)$ a polynomial. We show that $\mathrm{PSPACE} \subseteq \mathrm{NP}^{\mathrm{cR}_s^{CS}}$.

For the first two results we use the oblivious sampler construction of Zuckerman [29], a lemma [6] that measures the size of sets in terms of CD complexity, and we prove a lemma that shows that the first bits of a random string are in a sense more random than the whole string. For the last result we make use of the interactive protocol [22, 24] for quantified boolean formulas (QBFs).

To show optimality of our results for relativizing techniques, we construct an oracle world where our first result cannot be improved to deterministic reductions. We show that there is an oracle such that $\mathrm{BPP} \not\subseteq \mathrm{P}^{\mathrm{R}_t^{CD}}$ for any polynomial t. The construction of the oracle is an extension of the techniques developed by Beigel, Buhrman, and Fortnow [4].

2. Definitions and notations. We assume the reader is familiar with standard notions in complexity theory as can be found, e.g., in [2]. Strings are elements of Σ^*, where $\Sigma = \{0,1\}$. For a string s and integers $n, m \leq |s|$ we use the notation $s[n..m]$ for the string consisting of the nth through mth bit of s. We use λ for the empty string. We also need the notion of an *oblivious sampler* from [29].

DEFINITION 2.1. *A universal (r,d,m,ϵ,γ)-oblivious sampler is a deterministic algorithm which on input a uniformly random r-bit string outputs a sequence of points $z_1,\dots,z_d \in \{0,1\}^m$ such that for any collection of d functions $f_1,\dots,f_d : \{0,1\}^m \mapsto [0,1]$ it is the case that*

$$\Pr\left[\left|\frac{1}{d}\sum_{i=1}^{d}(f_i(z_i) - Ef_i)\right| \leq \epsilon\right] \geq 1-\gamma$$

(*where* $Ef_i = 2^{-m}\sum_{z\in\{0,1\}^m} f_i(z)$).

In our application of this definition, we will always use a single function f.

Fix a universal Turing machine U and a nondeterministic universal machine U_{N}. All our results are independent of the particular choice of universal machine. For the definition of Kolmogorov complexity we need the fact that the universal machine can, on input p, y, halt and output a string x. For the definition of distinguishing complexity below we need the fact that the universal machine on input p, x, y can either accept or reject. We also need resource bounded versions of this property.

We define the Kolmogorov complexity function $C(x|y)$ (see [20]) by $C(x|y) = \min\{|p| : U(p,y) = x\}$. We define unconditional Kolmogorov complexity by $C(x) = C(x|\lambda)$. Hartmanis [14] defined a time bounded version of Kolmogorov complexity, but resource bounded versions of Kolmogorov complexity date back as far as [3]. (See also [20].) Sipser [25] defined the distinguishing complexity CD^t.

We will need the following versions of resource bounded Kolmogorov complexity and distinguishing complexity.

- $CS^s(x|y) = \min\left\{ |p| : \begin{array}{ll} (1) & U(p,y) = x; \\ (2) & U(p,y) \text{ uses at most} \\ & s(|x|+|y|) \text{ tape cells} \end{array} \right\}.$

  (See [14].)
- $CD^t(x|y) = \min\left\{ |p| : \begin{array}{ll} (1) & U(p,x,y) \text{ accepts;} \\ (2) & (\forall z \neq x)U(p,z,y) \text{ rejects;} \\ (3) & (\forall z \in \Sigma^*)U(p,z,y) \text{ runs in at most} \\ & t(|u|+|y|) \text{ steps} \end{array} \right\}.$

  (See [25].)
- $CND^t(x|y) = \min\left\{ |p| : \begin{array}{ll} (1) & U_{\mathrm{N}}(p,x,y) \text{ accepts;} \\ (2) & (\forall z \neq x)U_{\mathrm{N}}(p,z,y) \text{ rejects;} \\ (3) & (\forall z \in \Sigma^*)U_{\mathrm{N}}(p,z,y) \text{ runs in at most} \\ & t(|z|+|y|) \text{ steps} \end{array} \right\}.$

  (See [6].)

For $0 < \epsilon \leq 1$ we define the following sets of strings of "maximal" CD^p and CND^p complexity.

- $\mathrm{R}_{t,\epsilon}^{CD} = \{x : CD^t(x|\lambda) \geq \epsilon|x|\}$.
- $\mathrm{R}_{t,\epsilon}^{CND} = \{x : CND^t(x|\lambda) \geq \epsilon|x|\}$.

Note that for $\epsilon = 1$ these sets are the sets mentioned in the introduction. In this case we will omit the ϵ and use R_t^{CD} and R_t^{CND}. We also define the set of strings of maximal space bounded complexity.

$$\mathrm{cR}_s^{CS} = \{\langle x, y\rangle : CS^s(x|y) \geq |x|\}.$$

The c in the notation is to emphasize that randomness is conditional. Also, cR_s^{CS} technically is a set of pairs rather than a set of strings. The unconditional space bounded random strings would be

$$\mathrm{R}_s^{CS} = \{x : \langle x, \lambda\rangle \in \mathrm{cR}_s^{CS}\}.$$

We have no theorems concerning this set.

The C complexity of a string is always upperbounded by its length plus some constant depending only on the choice of the universal machine. The CD and CND complexities of a string are always upperbounded by the C complexity of that string plus some constant depending again only on the particular choice of universal machine. All quantifiers used in this paper are polynomially bounded. Often the particular polynomial is not important for what follows, or it is clear from the context and is omitted. Sometimes we need explicit bounds. Then the particular bound is given as a superscript to the quantifier. For example, we use $\exists^m y$ to denote "there exists a y with $|y| \leq m$," or $\forall^{=n} x$ to denote "for all x of length n."

The classes MA and AM are defined as follows.

DEFINITION 2.2. *$L \in$ MA iff there exists a $|x|^c$ time bounded machine M such that*

1. $x \in L \implies \exists y \Pr[M(x, y, r) = 1] > 2/3$;
2. $x \notin L \implies \forall y \Pr[M(x, y, r) = 1] < 1/3$,

where r is chosen uniformly at random in $\{0,1\}^{|x|^c}$.

$L \in$ AM iff there exists a $|x|^c$ time bounded machine M such that

1. $x \in L \implies \Pr[\exists y M(x, y, r) = 1] > 2/3$;
2. $x \notin L \implies \Pr[\exists y M(x, y, r) = 1] < 1/3$,

where r is chosen uniformly at random in $\{0,1\}^{|x|^c}$.

It is known that NP $\cup$ BPP $\subseteq$ MA $\subseteq$ AM $\subseteq$ PSPACE [1].

Let $\#M(x)$ represent the number of accepting computations of a nondeterministic Turing machine M on input x. A language L is in $\oplus$P if there exists a polynomial time bounded nondeterministic Turing machine M such that for all x

- $x \in L \Rightarrow \#M(x)$ is odd;
- $x \notin L \Rightarrow \#M(x)$ is even.

Let g be any function. We say that advice function f is g-bounded if for all n it holds that $|f(n)| \leq g(n)$. In this paper we will be interested only in functions g that are polynomial.

The notation $\leq_T^{sn}$ is used for *strong nondeterministic Turing reductions*, which are defined by $A \leq_T^{sn} B$ iff $A \in \mathrm{NP}^B \cup \mathrm{CoNP}^B$.

3. Distinguishing complexity for derandomization. In this section we prove hardness of R_t^{CD} and R_t^{CND} for AM and MA games, respectively, under NP-reductions.

THEOREM 3.1. *For any t with $t(n) \in \omega(n \log n)$, MA $\subseteq \mathrm{NP}^{\mathrm{R}_t^{CD}}$.*

THEOREM 3.2. *For any t with $t(n) \in \omega(n)$, AM $\subseteq \mathrm{NP}^{\mathrm{R}_t^{CND}}$.*

The proof of both theorems is roughly as follows. First guess a string of high CD^{poly} complexity, respectively, CND^{poly} complexity. Next, we use the nondeterministic reductions once more to play the role of Merlin and use the random string to derandomize Arthur. Note that this is not as straightforward as it might look. The randomness used by Arthur in interactive protocols is used for hiding and cannot in general be substituted by computational randomness.

The idea of using strings of high CD complexity and Zuckerman's sampler derandomization stems from [7, section 8], which is the full version of [6]. Though they do not explicitly define the set R_t^{CD}, they use the same approach to derandomize BPP computations there.

The proof needs a string of high CD^p, respectively, CND^p complexity for p some polynomial. We first show that we can nondeterministically extract such a string from a longer string with high CD^t complexity (respectively, CND^t complexity) for any fixed t with $t(n) \in \omega(n \log n)$.

LEMMA 3.3. *Let f be such that $f(n) < n$, and let t, t', and T be such that $T(n) = (t'(f(n)) + n - f(n))$, $\lim_{n\mapsto\infty} \frac{T(n)\log T(n)}{t(n)} = 0$. Then for all sufficiently large s with $CD^t(s) > |s|$, it holds that $CD^{t'}(s[1..f(|s|)]) \geq f(|s|) - 2\log|f(|s|)| - O(1)$.*

Proof. Suppose for a contradiction that for any constant d_0 and infinitely many s with $CD^t(s) \geq |s|$, it holds that $CD^{t'}(s[1..f(|s|)]) < f(|s|) - 2\log|f(|s|)| - d_0$. Then for any such s there exists a program p_s that runs in $t'(f(|s|))$ and recognizes only $s[1..f(|s|)]$ where $|p_s| < f(|s|) - 2\log|f(|s|)| - d_0$. The following program then recognizes s and no other string.

> Input y.
> Check that the first $f(|s|)$ bits of y equal $s[1..f(|s|)]$, using p_s. (Assume $|f(|s|)|$ is stored in the program for a cost of $\log|f(|s|)|$ bits.)
> Check that the last $|s| - f(|s|)$ bits of y equal $s[f(|s|) + 1..|s|]$. (These bits are also stored in the program.)

This program runs in time $T(|s|) = t'(f(|s|)) + |s| - f(|s|)$. Therefore it takes at most $t(|s|)$ steps on U for all sufficiently large s [16]. We lose the $\log n$ factor here because our algorithm must run on a fixed machine and the simulation is deterministic.

The program's length is $|p_s| + |s| - f(|s|) + \log|f(|s|)| + d_1 < f(|s|) - 2\log|f(|s|)| - d_0 + |s| - f(|s|) + \log|f(|s|)| + d_1$, which is less than $|s|$ for almost all s. Hence $CD^t(s) < |s|$, which contradicts the assumption. □

COROLLARY 3.4. *For every polynomial n^c, $t \in \omega(n \log n)$ and sufficiently large string s with $CD^t(s) \geq |s|$, if $m = |s|^{\frac{1}{c}}$ and $s' = s[1..m]$, then $CD^{n^c}(s') \geq |s'| - 2\log|s'| - O(1)$.*

Proof. Take $t'(n) = n^c$, $f(n) = n^{\frac{1}{c}}$ and apply Lemma 3.3. □

Lemma 3.3 and Corollary 3.4 have the following nondeterministic analogon.

LEMMA 3.5. *For every polynomial n^c, $t \in \omega(n)$ and sufficiently large string s with $CND^t(s) \geq |s|$, if $m = |s|^{\frac{1}{c}}$ and $s' = s[1..m]$, then $CND^{n^c}(s') \geq |s'| - 2\log|s'| - O(1)$.*

Proof. The same proof applies, with a lemma similar to Lemma 3.3. However, in the nondeterministic case the simulation costs only linear time [5]. □

Before we can proceed with the proof of the theorems, we also need some earlier results. We first need the following theorem from Zuckerman.

THEOREM 3.6 (see [29]). *There is a constant c such that for $\gamma = \gamma(m)$, $\epsilon = \epsilon(m)$, and $\alpha = \alpha(m)$ with $m^{-1/2\log^* m} \leq \alpha \leq 1/2$ and $\epsilon \geq \exp(-\alpha^{2\log^* m} m)$, there exists a universal $(r, d, m, \epsilon, \gamma)$-oblivious sampler which runs in polynomial time and uses only $r = (1 + \alpha)(m + \log\gamma^{-1})$ random bits and outputs $d = ((m + log\gamma^{-1})/\epsilon)^{c_\alpha}$ sample points, where $c_\alpha = c(\log\alpha^{-1})/\alpha$.*

We also need the following lemma by Buhrman and Fortnow.

LEMMA 3.7 (see [6]). *Let A be a set in* P. *For each string $x \in A^{=n}$ it holds that $CD^p(x) \leq 2\log(\|A^{=n}\|) + O(\log n)$ for some polynomial p.*

As noted in [6], an analogous lemma holds for CND^p and NP.

LEMMA 3.8 (see [6]). *Let A be a set in* NP. *For each string $x \in A^{=n}$ it holds that $CND^p(x) \leq 2\log(\|A^{=n}\|) + O(\log n)$ for some polynomial p.*

From these results we can prove the theorems. If we want to prove, for Theorem 3.1, that an NP machine oracle with oracle R_t^{CD} can recognize a set A in MA, then the positive side of the proof is easy: If x is in A, then there exists a machine M and a string y such that a 2/3 fraction of the strings r of length $|x|^c$ makes $M(x, y, r)$ accept. So an NP machine can certainly guess one such pair x, y as a "proof" for $x \in A$. The negative side is harder. We will show that if $x \notin A$ and we substitute for r a string of high enough CD complexity (CND complexity for Theorem 3.2), then *no* y can make $M(x, y, r)$ accept.

To grasp the intuition behind the proof let us look at the much simplified example of a BPP machine M having a 1/3 error probability on input x and a string r of maximal unbounded Kolmogorov complexity. There are $2^{|x|^c}$ possible computations on input x, where $|x|^c$ is the runtime of M. Suppose that M must accept x. Then at most a 1/3 fraction, i.e., at most $2^{|x|^c}/3$ of these computations reject x. Each rejecting computation consists of a deterministic part described by M and x and a set of $|x|^c$ coin flips. Identify such a set of coin flips with a binary string and we have that each rejecting computation uniquely identifies a string of length $|x|^c$. Call this set B. We would like to show by contradiction that a random string cannot be a member of this set, and hence that any random string, used as a sequence of coin flips, leads to a correct result. Any string in B is described by M, x, and an index in B, which has length $\log\|B\| \leq |x|^c - \log 3$. So far there are no grounds for a contradiction since a description consisting of these elements can have length greater than $|x|^c$. However, we can *amplify* the computation of M on input x by repetition and taking majority. Suppose we repeat the computation $|x|^2$ times. This will increase the number of incorrect computations to (at most) $(\frac{8}{9})^{|x|^2/2}2^{|x|^{c+2}}$. An index in this set has length $|x|^{c+2} - (|x|^2/2)\log\frac{9}{8}$. However, $|x| + |x|^{c+2} - (|x|^2/2)\log\frac{9}{8}$ cannot describe a random string of length $|x|^{c+2}$, which is the length of such a computation.

Unfortunately, in our case the situation is a bit more complicated. The factor 2 in Lemma 3.7 renders standard amplifaction of randomized computation useless. Fortunately, Theorem 3.6 allows for a different type of amplification using much less random bits, so that the same type of argument *can* be used. We will now proceed to show how to fit the amplification given by Theorem 3.6 to our situation.

LEMMA 3.9.

1. *Let L be a language in* MA. *For any constant k and any constant $0 < \alpha \leq \frac{1}{2}$, there exists a deterministic polynomial time bounded machine M such that*

(a) $x \in L \implies \exists^m y \Pr[M(x, y, r) = 1] = 1$;

(b) $x \notin L \implies \forall^m y \Pr[M(x, y, r) = 1] < 2^{-km}$,

where $m = |x|^c$ and r is chosen uniformly at random in $\{0, 1\}^{(1+\alpha)(1+k)m}$.

2. *Let L be a language in* AM. *For any constant k and any constant $0 < \alpha \leq \frac{1}{2}$, there exists a deterministic polynomial time bounded machine M such that*

(a) $x \in L \implies \Pr[\exists y M(x, y, r) = 1] = 1$;

(b) $x \notin L \implies \Pr[\exists y M(x, y, r) = 1] < 2^{-km}$,

where $m = |x|^c$ and r is chosen uniformly at random in $\{0, 1\}^{(1+\alpha)(1+k)m}$.

Proof.

1. Fürer et al. showed that the fraction 2/3 (see Definition 2.2) can be replaced by 1 in [13]. Now let M_L be the deterministic polynomial time machine corresponding to L in Definition 2.2, adapted so that it can accept with probability 1 if $x \in L$. Assume

M_L runs in time n^c (where $n = |x|$). This means that for M_L the $\exists y$ and $\forall y$ in the definition can be assumed to be $\exists^{n^c} y$ and $\forall^{n^c} y$, respectively. Also, the random string may be assumed to be drawn uniformly at random from $\{0,1\}^{n^c}$.

To obtain the value 2^{-km} in the second item, we use Theorem 3.6 with $\gamma = 2^{-km}$, and $\epsilon = 1/6$. For given x and y let f_{xy} be the FP function that on input z computes $M_L(x, y, z)$. If $|y| = |z| = n^c = m$, then $f_{xy} : \{0,1\}^m \mapsto [0,1]$. We use the oblivious sampler to get a good estimate for Ef_{xy}. That is, we feed a random string of length $(1+\alpha)(1+k)m$ in the oblivious sampler and it returns $d = ((1+k)m/\epsilon)^{c_\alpha}$ sample points $z_1, \ldots, z_d$ on which we compute $\frac{1}{d}\sum_{i=1}^{d} f_{xy}(z_i)$. M is the machine that computes this sum on input x, y, and r and accepts iff its value is greater than $1/2$.

If $x \in L$, there is a y such that $\Pr[M_L(x, y, r) = 1] = 1$. This means $\frac{1}{d}\sum_{i=1}^{d} f_{xy}(z_i) = 1$ no matter which sample points are returned by the oblivious sampler. If $x \notin L$, then $(\forall y)[Ef_{xy} < 1/3]$. With probability $1 - \gamma$ the sample points returned by the oblivious sampler are such that $|\frac{1}{d}\sum_{i=1}^{d} f_{xy}(z_i) - Ef_{xy}| \le \epsilon$, so $\frac{1}{d}\sum_{i=1}^{d} f_{xy}(z_i) > \frac{1}{2}$ with probability $\le 2^{-km}$.

2. The proof is analogous to the proof of part 1. We just explain the differences. For the 1 in the first item of the claim we can again refer to [13]. In this part M_L is the deterministic polynomial time machine corresponding to the AM-language L and we define the function $f_x : \{0,1\}^m \mapsto [0,1]$ as the function that on input z computes $\exists^{n^c} y M_L(x, y, z) = 1$. Now f_x is an $\mathrm{FP}^{\mathrm{NP}}$ computable function. The sample points $z_1, \ldots, z_d$ that are returned in this case have the following properties. If $x \in L$, then $f_x(z_i) = 1$ no matter which string is returned as z_i. That is, for every possible sample point z_i, there is a y_i such that $M_L(x, y_i, z_i) = 1$. So for any set of sample points $z_1, \ldots, z_d$ that the sampler may return, there exists a $y = \langle y_1, \ldots, y_d \rangle$ such that $M_L(x, y_i, z_i) = 1\ \forall i$. If $x \notin L$, then $f_x(z_i) = 1$ for less than half of the sample points with probability $1 - \gamma$. That is,

$$\Pr\left[(\exists y = y_1 \ldots y_d)\left[\frac{1}{d}\sum_{i=1}^{d} M_L(x, y_i, z_i) > \frac{1}{2}\right]\right]$$

is less than 2^{-km}. So if we let $M(x, y, r)$ be the deterministic polynomial time machine that uses r to generate d sample points and then interprets y as $\langle y_1, \ldots, y_d \rangle$ and counts the number of accepts of $M_L(x, y_i, z_i)$ and accepts if this number is greater than $\frac{1}{2}d$, we get exactly the desired result. □

In the next lemma we show that a string of high enough CD^{poly} (CND^{poly}) can be used to *derandomize* an MA (AM) protocol.

LEMMA 3.10.

1. *Let L be a language in* MA *and $0 < \epsilon \le 1$. There exists a deterministic polynomial time bounded machine M, a polynomial q, $\alpha > 0$, and integers k and c such that for almost all n and every r with $|r| = (1+\alpha)(1+k)n^c$ and $CD^q(r) \ge \epsilon|r|$, $\forall^{=n} x[x \in L \iff \exists y M(x, y, r) = 1]$.*

2. *Let L be a language in* AM *and $0 < \epsilon \le 1$. There exists a deterministic polynomial time bounded machine M a polynomial q, $\alpha > 0$, and integers k and c such that for almost all n and every r with $|r| = (1+\alpha)(1+k)n^c$ and $CND^q(r) \ge \epsilon|r|$, $\forall^{=n} x[x \in L \iff \exists y M(x, y, r) = 1]$.*

Proof.

1. Choose $\alpha < \frac{\epsilon}{2}$ and $k > \frac{6}{\epsilon - 2\alpha}$. Let M be the deterministic polynomial time bounded machine corresponding to L, k, and α of Lemma 3.9(1). The polynomial n^c will be the time bound of the machine witnessing $L \in$ MA of that same lemma. We

will determine q later, but assume for now that r is a string of length $(1+\alpha)(1+k)n^c$ such that $CD^q(r) \geq \epsilon|r|$, and for ease of notation set $m = n^c$.

Suppose $x \in L$. Then it follows that there exists a y such that for all s of length $(1+\alpha)(1+k)n^c$, $M(x, y, s) = 1$. So in particular it holds that $M(x, y, r) = 1$.

Suppose $x \notin L$. We have to show that $(\forall y)[M(x, y, r) = 0]$. Suppose that this is not true and let y_0 be such that $M(x, y_0, r) = 1$. Define

$$A_{x,y_0} = \{s : M(x, y_0, s) = 1\}.$$

It follows that $A_{x,y_0} \in \mathrm{P}$ by essentially a program that simulates M and has x and y_0 hardwired. (Although A_{x,y_0} is finite and therefore trivially in P it is crucial here that the size of the polynomial program is roughly $|M| + |x| + |y_0|$.) Because of the amplification of the MA protocol we have that

$$\|A_{x,y_0}\| \leq 2^{(1+\alpha)(1+k)m-km}.$$

Since $r \in A_{x,y_0}$ it follows by Lemma 3.7 that there is a polynomial p such that

$$\begin{aligned} CD^p(r) &\leq 2[(1+\alpha)(1+k)m - km] + |x| \\ &\quad + |y_0| + O(\log m) \\ &\leq 2\alpha m + 2\alpha km + 5m. \end{aligned}$$

On the other hand, we chose r such that

$$\begin{aligned} CD^q(r) &\geq \epsilon|r| \\ &= (1+\alpha)(1+k)m\epsilon \\ &> 2\alpha m + 2\alpha km + 5m, \end{aligned}$$

which gives a contradiction for $q \geq p$.

2. Choose $\alpha < \frac{\epsilon}{2}$ and $k > \frac{5}{\epsilon - 2\alpha}$. Let M be the deterministic polynomial time bounded machine corresponding to L, α, and k of Lemma 3.9(2). Again, n^c will be the time bound of the machine now witnessing $L \in \mathrm{AM}$, $m = n^c$, and q will be determined later. Assume for now that r is a string of length $(1+\alpha)(1+k)n^c$ such that $CND^q(r) \geq \epsilon|r|$. Suppose $x \in L$. Then it follows that for all s there exists a y such that $M(x, y, s) = 1$. So in particular there is a y_r such that $M(x, y_r, r) = 1$. Suppose $x \notin L$. We have to show that $\forall y M(x, y, r) = 0$. Suppose that this is not true. Define $A_x = \{s : \exists y M(x, y, s) = 1\}$. Then $A_x \in \mathrm{NP}$ by a program that has x hardwired, guesses a y, and simulates M. Because of the amplification of the AM protocol we have that $\|A_x\| \leq 2^{(1+\alpha)(1+k)m-km}$. Since $r \in A_x$ it follows by Lemma 3.8 that there exists a polynomial p such that

$$\begin{aligned} CND^p(r) &\leq 2[(1+\alpha)(1+k)m - km] + |x| + O(\log m) \\ &\leq 2\alpha m + 2\alpha km + 4m. \end{aligned}$$

On the other hand, we chose r such that

$$\begin{aligned} CND^q(r) &\geq \epsilon|r| \\ &= (1+\alpha)(1+k)m\epsilon \\ &> 2\alpha m + 2\alpha km + 4m, \end{aligned}$$

which gives a contradiction whenever $q \geq p$. □

The following corollary shows that a string of high enough CD^{poly} complexity can be used to derandomize a BPP machine (see also [7, Theorem 8.2]).

COROLLARY 3.11. *Let A be a set in BPP. For any $\epsilon > 0$ there exists a polynomial time Turing machine M and a polynomial q such that if $CD^q(r) \geq \epsilon|r|$ with $|r| = q(n)$, then for all x of length n it holds that $x \in A \iff M(x,r) = 1$.*

Proof of Theorem 3.1. Let A be a language in MA. Let q, M, and $q'(n) = (1+\alpha)(1+k)q(n)$ be as in Lemma 3.10(1). The nondeterministic reduction behaves as follows on input x of length n. First guess an s of size $q(q'(n))$ and check that $s \in \mathrm{R}_t^{CD}$. Set $r = s[1..q'(n)]$ and accept iff there exists a y such that $M(x,y,r) = 1$. By Corollary 3.4 it follows that $CD^q(r) \geq |r|/2$ and the correctness of the reductions follows directly from Lemma 3.10(1) with $\epsilon = 1/2$. □

Proof of Theorem 3.2. This follows directly from Lemma 3.10(2). The NP-algorithm is analogous to the one above. □

COROLLARY 3.12. *For $t \in \omega(n \log n)$*

1. BPP *and* $\mathrm{NP}^{\mathrm{BPP}}$ *are included in* $\mathrm{NP}^{\mathrm{R}_t^{CD}}$;
2. $\overline{GI} \in \mathrm{NP}^{\mathrm{R}_t^{CND}}$.

It follows that if $\mathrm{R}_t^{CND} \in \mathrm{NP} \cap \mathrm{coNP}$, then the graph isomorphism (GI) problem is in $\mathrm{NP} \cap \mathrm{coNP}$.

4. Limitations. In the previous section we showed that the set R_t^{CD} is hard for MA under NP reductions. One might wonder whether R_t^{CD} is also hard for MA under a stronger reduction like the deterministic polynomial time Turing reduction. In this section we show that, if true, this will need a nonrelativizing proof. We will derive the following theorem.

THEOREM 4.1. *There is a relativized world where for every polynomial t and $0 < \epsilon \leq 1$, $\mathrm{BPP} \not\subseteq \mathrm{P}^{\mathrm{R}_{t,\epsilon}^{CD}}$.*

The proof of this theorem is given in Lemma 4.2, which says that the statement of Theorem 4.1 is true in any world where $\mathrm{P}^A = \oplus\mathrm{P}^A$ and $\mathrm{EXP}^{\mathrm{NP}^A} \subseteq \mathrm{NP}^A/poly$, and in Theorem 4.3, which precisely shows the existence of such a world.

LEMMA 4.2. *For any oracle A and $0 < \epsilon \leq 1$ it holds that if $\mathrm{EXP}^{\mathrm{NP}^A} \subseteq \mathrm{NP}^A/poly$ and $\oplus\mathrm{P}^A = \mathrm{P}^A$, then $\mathrm{BPP}^A \not\subseteq P^{\mathrm{R}_{t,\epsilon}^{CD\,A}}$.*

Proof. Suppose for a contradiction that the lemma is not true. If $\mathrm{EXP}^{\mathrm{NP}} \subseteq \mathrm{NP}/poly$, then $\mathrm{EXP} \subseteq \mathrm{NP}/poly$, so $\mathrm{EXP} \subseteq \mathrm{PH}$ [27]. Furthermore, if $\mathrm{EXP}^{\mathrm{NP}} \subseteq \mathrm{NP}/poly$, then certainly $\mathrm{EXP}^{\mathrm{NP}} \subseteq \mathrm{EXP}/poly$. It then follows from [9] that $\mathrm{EXP}^{\mathrm{NP}} = \mathrm{EXP}$, so $\mathrm{EXP}^{\mathrm{NP}} \subseteq \mathrm{PH}$.

If $\oplus\mathrm{P} = \mathrm{P}$, then unique-SAT (see [8] for a definition) is in P. Then $\mathrm{NP} = \mathrm{R}$ by [26] and so $\mathrm{NP} \subseteq \mathrm{BPP}$ which implies $\mathrm{PH} \subseteq \mathrm{BPP}$ by [28].

Finally, the fact that unique-SAT is in P is equivalent to the following: for all x and y, $C^{poly}(x|y) \leq CD^{poly}(x|y) + O(1)$, as shown in [12]. We can use the proof of [12] to show that unique-SAT in P also implies that $\mathrm{R}_{t,\epsilon}^{CD} \in \mathrm{coNP}$ for a particular universal machine. (Note that we need only contradict the assumption for one particular type of universal machine.) This then in its turn implies by assumption that BPP and hence $\mathrm{EXP}^{\mathrm{NP}}$ are in P^{NP}. This, however, contradicts the hierarchy theorem for relativized Turing machines [15]. As all parts of this proof relativize, we get the result for any oracle. There's one caveat here. Though $\mathrm{R}_{t,\epsilon}^{CD\,A}$ clearly has a meaningful interpretation, to talk about $\mathrm{P}^{\mathrm{R}_{t,\epsilon}^{CD\,A}}$ one must of course allow P to have access to the oracle. It is not clear that P can ask any question if the machine can only ask a question about the random strings. Therefore, one might argue that $\mathrm{P}^{\mathrm{R}_{t,\epsilon}^{CD\,A} \oplus A}$ should actually be in the statement of the lemma. This does not affect the proof.

Our universal machine, say, U_S, is the following. On input p, x, y, U_S uses the Cook–Levin reduction to produce a formula f on $|x|$ variables with the property that x satisfies f iff p accepts x. Then U_S uses the self-reducibility of f and the assumed polynomial time algorithm for unique-SAT to make acceptance of x unique. That is, first if the number of variables is not equal $|y|$, it rejects. Then, using the well-known substitute and reduce algorithm for SAT, it verifies for $i = 1, \ldots, |x|$ and assignments $x_j = v_j$ successively obtained from the algorithm that the algorithm for unique-SAT precisely accepts $f(v_1 \ldots v_i)$ or rejects if this algorithm accepts both $f(v_1 \ldots v_i)$ and $f(v_1 \ldots (1 - v_i))$. Using this universal machine every program accepts at most one string and therefore $\mathrm{R}^{CD}_{t,\epsilon} \in$ coNP via an obvious predicate. As argued above, this gives us our contradiction. $\square$

Now we proceed to construct the oracle.

THEOREM 4.3. *There exists an oracle A such that* $\mathrm{EXP}^{\mathrm{NP}^A} \subset \mathrm{NP}^A/poly \wedge \oplus\mathrm{P}^A = \mathrm{P}^A$.

Proof. The proof parallels the construction from Beigel, Buhrman, and Fortnow [4], who construct an oracle such that $\mathrm{P}^A = \oplus\mathrm{P}^A$ and $\mathrm{NEXP}^A = \mathrm{NP}^A$. We will use a similar setup.

Let M^A be a nondeterministic *linear* time Turing machine such that the language L^A defined by

$$w \in L^A \Leftrightarrow \#M^A(w) \bmod 2 = 1$$

is $\oplus P^A$ complete for every A.

For every oracle A, let K^A be the linear time computable complete set for NP^A. Let N^{K^A} be a deterministic machine that runs in time 2^n and for every A accepts a language H^A that is complete for $\mathrm{EXP}^{\mathrm{NP}^A}$. We will construct A such that there exists a n^2 bounded advice function f such that for all w

$$\begin{array}{lcll} w \in L^A & \Leftrightarrow & \langle 0, w, 1^{|w|^2} \rangle \in A & \text{(Condition 0)}, \\ w \in H^A & \Leftrightarrow & \exists v\ |v| = |w|^2 \text{ and} & \\ & & \langle 1, f(|w|), w, v \rangle \in A & \text{(Condition 1)}. \end{array}$$

Condition 0 will guarantee that $\mathrm{P} = \oplus\mathrm{P}$, and Condition 1 will guarantee that $\mathrm{EXP}^{\mathrm{NP}} \subset \mathrm{NP/poly}$.

We use the term 0-strings for the strings of the form $\langle 0, w, 1^{|w|^2} \rangle$ and 1-strings for the strings of the form $\langle 1, z, w, v \rangle$ with $|z| = |v| = |w|^2$. All other strings we immediately put in $\overline{A}$.

First we give some intuition for the proof. M is a linear time Turing machine. Therefore setting the 1-strings forces the setting of the 0-strings. Condition 0 will be automatically fulfilled by just describing how we set the 1-strings because they force the 0-strings as defined by Condition 0.

Fulfilling Condition 1 requires a bit more care since $N^{K^A}(x)$ can query exponentially long and *double* exponentially many 0- and 1-strings. We consider each 1-string $\langle 1, z, w, v \rangle$ as a 0-1 valued variable $y_{\langle z,w,v \rangle}$ whose value determines whether $\langle 1, z, w, v \rangle$ is in A. The construction of A will force a 1-1 correspondence between the computation of $N^{K^A}(x)$ and a low-degree polynomial over variables with values in $GF[2]$. To encode the computation properly we use the fact that the OR function has high degree.

We will assign a polynomial p_z over GF[2] to all of the 0-strings and 1-strings z. We ensure that for all z

1. if $p_z = 1$, then z is in A;
2. if $p_z = 0$, then z is not in A.

First for each 1-string $z = \langle 1, z, w, v\rangle$ we let p_z be the single variable polynomial $y_{\langle z,w,v\rangle}$.

We assign polynomials to the 0-strings recursively. Note that $M^A(x)$ can only query 0-strings with $|w| \leq \sqrt{|x|}$. Consider an accepting computation path π of $M(x)$ (assuming the oracle queries are guessed correctly). Let $q_{\pi,1}, \ldots, q_{\pi,m}$ be the queries on this path and $b_{\pi,1}, \ldots, b_{\pi,m}$ be the query answers with $b_{\pi,i} = 1$ if the query was guessed in A, and $b_{\pi,i} = 0$ otherwise. Note that $m \leq n = |x|$.

Let $\mathcal{P}$ be the set of accepting computation paths of $M(x)$. We then define the polynomial p_z for $z = \langle 0, x, 1^{|x|^2}\rangle$ as follows:

$$p_z = \sum_{\pi\in\mathcal{P}} \prod_{1\leq i\leq m} (p_{q_{\pi,i}} + b_{\pi,i} + 1). \tag{1}$$

Remember that we are working over GF[2] so addition is parity.

Setting the variables $y_{\langle z,w,v\rangle}$ (and thus the 1-strings) forces the values of p_z for the 0-strings. We have set things up properly so the following lemma is straightforward.

LEMMA 4.4. *For each* 0*-string* $z = \langle 0, x, 1^{|x|^2}\rangle$ *we have* $p_z = \#M^A(x) \bmod 2$ *and Condition* 0 *can be satisfied. The polynomial* p_z *has degree at most* $|x|^2$.

Proof. The proof is simple by induction on $|x|$. □

The construction will be done in stages. At stage n we will code all the strings of length n of H^A into A setting some of the 1-strings and automatically the 0-strings and thus fulfilling both Conditions 0 and 1 for this stage.

We will need to know the degree of the multivariate multilinear polynomials representing the *OR* and the *AND* function.

LEMMA 4.5. *The representation of the function* $OR(u_1, \ldots, u_m)$ *and the function* $AND(u_1, \ldots, u_m)$ *as multivariate multilinear polynomials over* GF[2] *requires degree exactly* m.

Proof. Every function over GF[2] has a unique representation as a multivariate multilinear polynomial.

Note that *AND* is just the product and by using De Morgan's laws we can write *OR* as

$$OR(u_1, \ldots, u_m) = 1 + \prod_{1\leq i\leq m} (1 + u_i). \qquad \square$$

The construction of the oracle now treats all strings of length n in lexicographic order. First, in a *forcing phase* in which the oracle is set so that all computations of N^{K^A} remain fixed for future extensions of the oracle, and then in a *coding phase* in which first an advice string is picked and then the computations just forced are coded in the oracle in such a way that they can be retrieved by an NP machine with this advice string. Great care has of course to be taken so that the two phases don't disturb each other and do not disturb earlier stages of the construction.

We first describe the *forcing phase*. Without loss of generality, we will assume that machine N queries only strings of the form $q \in K^A$. Note that since N runs in time 2^n it may query exponentially long strings to K^A.

Let x_1 be the first string of length n. When we examine the computation of $N(x_1)$ we encounter the first query q_1 to K^A. We will try to extend the oracle A to $A' \supseteq A$ such that $q_1 \in K^{A'}$. If such an extension does not exist we may assume that q_1 will *never* be in K^A no matter how we extend A in the future. We must, however, take care that we will not disturb previous queries that were forced to be in K^A. To this end we will build a set S containing all the previously encountered queries that were forced to be in K^A. We will only extend A such that $\forall q \in S$ it holds that $q \in K^{A'}$. We will call such an extension an *S-consistent* extension of A.

Returning to the computation of $N(x_1)$ and q_1 we ask whether there is an S-consistent extension of A such that $q_1 \in K^{A'}$. If such an extension exists, we will choose the S-consistent extension of A which adds a minimal number of strings to A and puts q_1 in S. Next we continue the computation of $N^{K^A}(x_1)$ with q_1 answered yes, and otherwise we continue with q_1 answered no. The next lemma shows that a minimal extension of A will never add more than 2^{3n} strings to A.

LEMMA 4.6. *Let S be as above and q be any query to K^A and suppose we are in stage n. If there exists an S-consistent extension of A such that $q \in K^{A'}$, then there exists one that adds at most 2^{3n} strings to A.*

Proof. Let M_K be a machine that accepts K^A when given oracle A and consider the computation of machine $M_K^A(q)$. Let $o_1, \ldots, o_l$ be the smallest set of strings such that adding them to A is an S-consistent extension of A such that $M_K^{A'}(q)$ accepts. ($A' = A \cup \{o_1, \ldots, o_l\}$.) Consider the leftmost accepting path of $M_K^{A'}(q)$ and let $q_1, \ldots, q_{2^n}$ be the queries (both 0- and 1-queries) on that path. Moreover let b_i be 1 iff $q_i \in A'$. Define for q the following polynomial:

$$P_q = \prod_{1 \le i \le 2^n} (p_{q_i} + b_i + 1). \tag{2}$$

After adding the strings $o_1, \ldots, o_l$ to A we have that $P_q = 1$. Moreover by Lemma 4.4 the degree of each p_{q_i} is at most 2^{2n} and hence the degree of P_q is at most 2^{3n}. Now consider what happens when we take out any number of the strings $o_1, \ldots, o_l$ of A' resulting in A''. Since this was a minimal extension of A it follows that $M_K^{A''}(q)$ rejects and that $P_q = 0$. So P_q computes the *AND* on the l strings $o_1, \ldots, o_l$. Since by Lemma 4.5 the degree of the unique multivariate multilinear polynomial that computes the *AND* over l variables over GF[2] is l, it follows that $l \le 2^{3n}$. □

After we have dealt with all the queries encountered on $N^{K^A}(x_1)$ we continue this process with the other strings of length n in lexicographic order. Note that since we only extend A S-consistently we will never disturb any computation of N^{K^A} on lexicographic smaller strings. This follows since the queries that are forced to be yes will remain yes, and the queries that could not be forced with an S-consistent extension will never be forced by any S'-consistent extension of A for $S \subset S'$. After we have finished this process we have to code all the computations of N on the strings of length n. It is easy to see that $\|S\| \le 2^{2n}$ and that at this point by Lemma 4.6 at most 2^{5n} strings have been added to A at this stage. Closing the *forcing phase* we can now pick an advice string and proceed to the *coding phase.* A standard counting argument shows that there is a string z of length n^2 such that no strings of the form $\langle 1, z, w, v\rangle$ have been added to A. This string z will be the advice for strings of length n.

Now we have to show that we can code every string x of length n correctly in A to fulfill Condition 1. We will do this in lexicographic order. Suppose we have coded all strings x_j (for $j < i$) correctly and that we want to code x_i. There are two cases.

Case 1. $N^{K^A}(x_i) = 0$. In this case we put all the strings $\langle 1, z, x_i, w\rangle$ in $\overline{A}$ and thus set all these variables to 0. Since this does not change the oracle it is an S-consistent extension.

Case 2. $N^{K^A}(x_i) = 1$. We properly extend A S-consistently adding only strings of the form $\langle 1, z, x_i, w\rangle$ to A. The following lemma shows that this can always be done. A *proper* extension of A is one that adds one or more strings to A.

LEMMA 4.7. *Let $\|S\| \le 2^{2n}$ be as above. Suppose that $N^{K^A}(x_i) = 1$. There exists a proper S-consistent extension of A adding only strings of the form $\langle 1, z, x_i, w\rangle$ with $|w| = n^2$.*

Proof. Suppose that no such proper S-consistent extension of A exists. Consider the following polynomial:

$$Q_{x_i} = 1 - \prod_{q \in S} (P_q), \tag{3}$$

where P_q is defined as in Lemma 4.6, equation (2). Initially $Q_{x_i} = 0$ and the degree of $Q_{x_i} \le 2^{5n}$. Since every extension of A with strings of the form $\langle 1, z, x_i, w\rangle$ is not S-consistent it follows that Q_{x_i} computes the OR of the variables $y_{\langle z, x_i, w\rangle}$. Since there are 2^{n^2} many of those variables we have by Lemma 4.5 a contradiction with the degree of Q_{x_i}. Hence there exists a proper S-consistent extension of A adding only strings of the form $\langle 1, z, x_i, w\rangle$, and x_i is properly coded into A. □

Stage n ends after coding all the strings of length n.

This completes the proof of Theorem 4.3. □

Theorem 4.3 together with the proof of Lemma 4.2 also gives the following corollary.

COROLLARY 4.8. *There exists a relativized world where* $\mathrm{EXP}^{\mathrm{NP}}$ *is in* BPP *and* $\oplus\mathrm{P} = \mathrm{P}$.

Our oracle also extends the oracle of Ko [17] to CD^{poly} complexity as follows.

COROLLARY 4.9. *There exists an oracle such that $\overline{\mathrm{R}^{CD}_{t,\epsilon}}$ for any $t \in \omega(n \log(n))$ and $\epsilon > 0$ is complete for* NP *under strong nondeterministic reductions and* $\mathrm{P}^{\mathrm{NP}} \ne \Sigma_2^p$.

Proof. The relativized world constructed in the proof of Theorem 4.3 is a world where $\overline{\mathrm{coNP}} \subseteq \mathrm{BPP}$ and $C^{poly}(x|y) = CD^{poly}(x|y) + O(1)$. Hence it follows that $\overline{\mathrm{R}^{CD}_{t,\epsilon}} \in \mathrm{NP}$. Moreover Corollary 3.12 relativizes so by item 1 we have that $\mathrm{BPP} \subseteq \mathrm{NP}^{\overline{\mathrm{R}^{CD}_{t,\epsilon}}}$. □

As a by-product our oracle shows the following.

COROLLARY 4.10. $\exists A$ *unique-SAT* $^A \in \mathrm{P}^A$ *and* $\mathrm{P}^{\mathrm{NP}^A} \ne \Sigma_2^{p,A}$.

This corollary indicates that the current proof that shows that if unique-SAT $\in$ P, then PH $= \Sigma_2^p$ cannot be improved to yield a collapse to P^{NP} using relativizing techniques.

5. PSPACE and $\mathbf{cR}_s^{CS}$. In this section we further study the connection between cR_s^{CS} and interactive proofs. So far we have established that strings that have sufficiently high CND^{poly} complexity can be used to derandomize an IP protocol that has two rounds in such a way that the role of both the prover and the verifier can be played by an NP oracle machine. Here we will see that this is also true for IP itself provided that the random strings have high enough space bounded Kolmogorov complexity. The set of QBFs is defined as the closure of the set of boolean variables x_i and their negations $\overline{x_i}$ under the operations $\wedge$, $\vee$, $\forall x_i$, and $\exists x_i$. A QBF in which all the variables are quantified is called *closed*. Other QBFs are called open. We need the following definitions and theorems from [24].

DEFINITION 5.1 (see [24]). *A QBF B is called* simple *if in the given syntactic representation every occurrence of each variable is separated from its point of quantification by at most one universal quantifier (and arbitrarily many other symbols).*

For technical reasons we also assume that (simple) QBFs can contain negated variables, but no other negations. This is no loss of generality since negations can be pushed all the way down to variables.

DEFINITION 5.2 (see [24]). *The arithmetization of a (simple) QBF B is an arithmetic expression obtained from B by replacing every positive occurrence of x_i by variable z_i, every negated occurrence of x_i by $(1 - z_i)$, every $\wedge$ by $\times$, every $\vee$ by $+$, every $\forall x_i$ by $\prod_{z_i \in \{0,1\}}$, and every $\exists x_i$ by $\sum_{z_i \in \{0,1\}}$.*

It follows that the arithmetization of a (simple) QBF in closed form has an integer value, whereas the arithmetization of an open QBF is equivalent to a (possibly multivariate) function.

DEFINITION 5.3 (see [24]). *The functional form of a simple closed QBF is the univariate function that is obtained by removing from the arithmetization of B either $\sum_{z_i \in \{0,1\}}$ or $\prod_{z_i \in \{0,1\}}$ where i is the least index of a variable for which this is possible.*

Notation. Let B be a (simple) QBF with quantifiers $Q_1, \ldots, Q_k$. For $i \leq k$ we let $*_i = +$ if $Q_i = \exists$ and $*_i = \times$ if $Q_i = \forall$. Let B be a QBF. Let B' be the boolean formula obtained from B by removing all its quantifiers. We denote by $\tilde{B}$ the arithmetization of B'. It is well known that the language of all true QBFs is complete for PSPACE. The restriction of true QBFs to simple QBFs remains complete.

THEOREM 5.4 (see [24]). *The language of all closed simple true QBFs is complete for* PSPACE *(under polynomial time many-one reductions).*

It is straightforward that the arithmetization of a QBF takes on a positive value iff the QBF is true. This fact also holds relative a not-too-large prime.

THEOREM 5.5 (see [24]). *A simple closed QBF B is true iff there exists a prime number P of size polynomial in $|B|$ such that the value of the arithmetization of B is positive modulo P. Moreover if B is false, then the value of the arithmetization of B is* 0 *modulo* any *such prime.*

THEOREM 5.6 (see [24]). *The functional form of every simple QBF can be represented by a univariate polynomial of degree at most* 3.

THEOREM 5.7 (see [24]). *For every simple QBF there exists an interactive protocol with prover P and polynomial time bounded verifier V such that*

1. *when B is true and P is honest, V always accepts the proof;*
2. *when B is false, V accepts the proof with negligible probability.*

The proof of Theorem 5.7 essentially uses Theorem 5.6 to translate a simple QBF to a polynomial in the following way. First, the arithmetization of a simple QBF B in closed form is an integer value V which is positive iff B is true. Then B's functional form F (recall that this is arithmetization of the QBF that is obtained from B by deleting the first quantifier) is a univariate polynomial p_1 of degree at most 3 which has the property that $p_1(0) *_1 p_1(1) = V$. Substituting any value r_1 in p_1 gives a new integer value V_1, which is of course the same value that we get when we substitute r_1 in F. However, $F(r_1)$ can again be converted to a (low-degree) polynomial by deleting its first $\sum$ or $\prod$ sign, and the above game can be repeated. Thus, we obtain a sequence of polynomials. From the first polynomial in this sequence V can be computed. The last polynomial p_n has the property that $p_n(r_1, \ldots, r_n) = \tilde{B}(r_1, \ldots, r_n)$. Two more things are needed: First, if any *other* sequence of polynomials $q_1, \ldots, q_n$ has the property that $q_1(0) *_1 q_1(1) \neq V$, $q_{i+1}(0) *_{i+1} q_{i+1}(1) = q_i(r_i)$, and $q_n(r_n) = \tilde{B}(r_1, \ldots, r_n)$, then there has to be some i where $q_i(r_i) = p_i(r_i)$, yet $q_i \neq p_i$. That is, r_i is an intersection point of p_i and q_i. Second, all calculations can be done modulo some prime number of polynomial size (Theorem 5.5). We summarize this in the following observation, which is actually a skeleton of the proof of Theorem 5.7.

OBSERVATION 5.8 (see [24, 22]). *Let B be a closed simple QBF wherein the quantifiers are $Q_1, \dots Q_n$ if read from left to right in its syntactic representation. Let A be its arithmetization, and let V be the value of A. There exist a prime number P of size polynomial in $|B|$ such that for any sequence $r_1, \dots, r_n$ of numbers taken from $[1..P]$ there is a sequence of polynomials of degree at most 3 and size polynomial in $|B|$ such that*

1. $p_1(0) \ast_1 p_1(1) = V$ *and* $V > 0$ *iff B is true;*
2. $p_{i+1}(0) \ast_{i+1} p_{i+1}(1) = p_i(r_i)$;
3. $p_n(r_n) = \tilde{B}(r_1, \dots, r_n)$;
4. *for any sequence of univariate polynomials $q_1, \dots, q_n$ such that*
   (a) $p_1(0) \ast_1 p_1(1) \neq q_1(0) \ast_1 q_1(1)$ *and*
   (b) $q_{i+1}(0) \ast_{i+1} q_{i+1}(1) = q_i(r_i)$ *and*
   (c) $q_n(r_n) = \tilde{B}(r_1, \dots, r_n)$,
   there is a minimal i such that $p_i \neq q_i$, yet $p_i(r_i) = q_i(r_i)$. That is, r_i is an intersection point of p_i and q_i.

Where all (in)equalities hold modulo P and hold modulo any *prime of polynomial size if B is false. Moreover, p_i can be computed in space $(|B| + |P|)^2$ from B, P, and $r_1, \dots, r_{i-1}$.*

From this reformulation of Theorem 5.7 we obtain that for any sequence of univariate polynomials $q_1, \dots, q_n$ and sequence of values $r_1, \dots, r_n$ that satisfy items 2 and 3 in Observation 5.8 it holds that either $q_1(0) \ast_1 q_1(1)$ is the true value of the arithmetization of B, or there is some polynomial q_i in this sequence such that r_i is an intersection point of p_i and q_i (where p_i is as in Observation 5.8). As p_i can be computed in quadratic space from B, P, and $r_1, \dots, r_{i-1}$ it follows that in the latter case r_i cannot have high space bounded Kolmogorov complexity relative to B, P, $q_1, \dots, q_i$, $r_1, \dots, r_{i-1}$. Hence, if r_i *does* have high space bounded Kolmogorov complexity, then r_i is *not* an intersection point, so the first case must hold (i.e., the value computed from q_1 is the true value of the arithmetization of B). The following lemma makes this precise.

LEMMA 5.9. *Assume the following for B, P, n, $q_1, \dots, q_n$, $r_1, \dots, r_n$, and $y_1, \dots, y_n$.*

1. *B is a simple* false *closed QBF on n variables.*
2. *P is a prime number $\geq 2^{|B|}$ of size polynomial in $|B|$.*
3. *$q_1 \dots q_n$ is a sequence of polynomials of degree 3 with coefficients in $[1..P]$.*
4. *$r_1, \dots, r_n$ are numbers in $[1..P]$.*
5. $y_1 = B\#P\#q_1\# \dots \#q_n$ *and* $y_{i+1} = y_i\#r_i$.
6. $CS^{n^2}(r_i \mid y_i) \geq |P|$.
7. $(\forall i \geq 2)[q_{i-1}(r_{i-1}) = q_i(0) \ast_i q_i(1) \mod P]$.
8. $\tilde{B}(r_1, \dots, r_n) = q_n(r_n) \mod P$.

Then $q_1(0) \ast_1 q_1(1) = 0 \mod P$.

Proof. Take all calculations modulo P. Suppose $q_1(0) \ast_1 q_1(1) \neq 0$. It follows from Observation 5.8 that there exists a sequence $p_1, \dots, p_n$ satisfying items 1 through 3 of that lemma. Furthermore since B is false $p_1(0) \ast_1 p_1(1) = 0$ modulo *any* prime, so $p_1(0) \ast_1 p_1(1) \neq q_1(0) \ast_1 q_1(1)$. It follows that there must be a minimal i such that $p_i \neq q_i$ and r_i is an intersection point of p_i and q_i. However, p_i can be computed in space $(|B| + |P|)^2$ from B, P, and $r_1, \dots, r_{i-1}$. As both p_i and q_i have degree at most

3, it follows that $CS^{n^2}(r_i \mid y_i)$ is bounded by a constant—a contradiction. □

This suffices for the main theorem of this section. Let s be any polynomial.

THEOREM 5.10. $\mathrm{PSPACE} \subseteq \mathrm{NP}^{\mathrm{cR}_s^{CS}}$.

Proof. We prove the lemma for $s(n) = n^2$, but the proof can by padding be extended to any polynomial. There exists an NP oracle machine that accepts the language of all simple closed true QBFs as follows. On input B first check that B is simple. Guess a prime number $P \geq 2^{|B|}$ of size polynomial in $|B|$, a sequence of polynomials $p_1, \ldots, p_n$ of degree at most 3 and with coefficients in $[1..P]$. Finally guess a sequence of numbers $r_1, \ldots, r_n$ all of size $|P|$. Check that

1. $p_1(0) \circledast_1 p_1(1) > 0$ and
2. $p_{i+1}(0) \circledast_{i+1} p_{i+1}(1) = p_i(r_i)$ and
3. $p_n(r_n) = \tilde{B}(r_1, \ldots, r_n)$ and
4. finally that $(\forall i \leq n)[CS^{n^2}(r_i|y_i) \geq |P|]$.

If B is true, Lemma 5.8 guarantees that these items can be guessed such that all tests are passed. If B is false and no other test fails, then Lemma 5.9 guarantees that $p_1(0) \circledast_1 p_1(1) = 0$, so the first check must fail. □

By the fact that PSPACE is closed under complement and the fact that cR_s^{CS} is also in PSPACE Theorem 5.10 gives that cR_s^{CS} is complete for PSPACE under strong nondeterministic reductions [21].

COROLLARY 5.11. cR_s^{CS} *is complete for* PSPACE *under strong nondeterministic reductions.*

Buhrman and Mayordomo [10] showed that for $t(n) = 2^{n^k}$, the set $R_t^C = \{x : C^t(x) \geq |x|\}$ is *not* hard for EXP under deterministic Turing reductions. In Theorem 5.10 we made use of the relativized Kolmogorov complexity (i.e., $CS^s(x|y)$). Using exactly the same proof as in [10] one can prove that the set $\mathrm{cR}_t^C = \{\langle x, y\rangle : C^t(x|y) \geq |x|\}$ is not hard for EXP under Turing reductions. On the other hand the proof of Theorem 5.10 also works for this set: $\mathrm{PSPACE} \subseteq \mathrm{NP}^{\mathrm{cR}_t^C}$. We suspect that it is possible to extend this to show that $\mathrm{EXP} \subseteq \mathrm{NP}^{\mathrm{cR}_t^C}$. So far, we have been unable to prove this.

Acknowledgments. We thank Paul Vitányi for interesting discussions and providing the title of this paper. We also thank two anonymous referees, who helped with a number of technical issues that cleared up much of the proofs and who pointed us to more correct references. One of the referees also pointed out Corollary 4.8.

REFERENCES

[1] L. BABAI, *Trading group theory for randomness*, in Proceedings of the 17th ACM Symposium on Theory of Computing, Providence, RI, 1985, pp. 421–429.

[2] J. BALCÁZAR, J. DÍAZ, AND J. GABARRÓ, *Structural Complexity* I, Springer-Verlag, Berlin, 1988.

[3] J. M. BARZDIN, *Complexity of programs to determine whether natural numbers not greater than n belong to a recursively enumerable set*, Dokl. Akad. Nauk SSSR, 9 (1968), pp. 1251–1254.

[4] R. BEIGEL, H. BUHRMAN, AND L. FORTNOW, *NP might not be as easy as detecting unique solutions*, in Proceedings of the 30th ACM Symposium on Theory of Computing, Dallas, TX, ACM, New York, 1998, pp. 203–208.

[5] R. BOOK, S. GREIBACH, AND B. WEGBREIT, *Time- and tape-bound Turing acceptors and afl's*, J. Comput. System Sci., 4 (1970), pp. 606–621.

[6] H. BUHRMAN AND L. FORTNOW, *Resource bounded Kolmogorov complexity revisited*, in Proceedings of the 14th Annual Symposium on Theoretical Computer Science, Lecture Notes in Comput. Sci. 1200, Springer-Verlag, Berlin, 1997, pp. 105–116.

[7] H. BUHRMAN AND L. FORTNOW, *Resource Bounded Kolmogorov Complexity Revisited*, manuscript, available from http://www.neci.nj.nec.com/homepages/fortnow/.

[8] H. Buhrman, L. Fortnow, and L. Torenvliet, *Six hypotheses in search of a theorem*, in Proceedings of the 12th Annual IEEE Conference on Computational Complexity, Ulm, Germany, 1997, pp. 2–12.

[9] H. Buhrman and S. Homer, *Superpolynomial circuits, almost sparse oracles and the exponential hierarchy*, in Proceedings of the 12th Conference on the Foundations of Software Technology and Theoretical Computer Science, Lecture Notes in Comput. Sci. 652, R. Shyamasundar, ed., Springer-Verlag, Berlin, 1992, pp. 116–127.

[10] H. Buhrman and E. Mayordomo, *An excursion to the kolmogorov random strings*, in Proceedings of the 10th Annual Conference on Structure in Complexity Theory, Minneapolis, MN, 1995, IEEE Computer Society Press, Los Alamitos, CA, pp. 197–205.

[11] H. Buhrman and L. Torenvliet, *Complete sets and structure in subrecursive classes*, in Lecture Notes Logic 12, Springer-Verlag, Berlin, 1998, pp. 45–78.

[12] L. Fortnow and M. Kummer, *Resource-bounded instance complexity*, Theoret. Comput. Sci. A, 161 (1996), pp. 123–140.

[13] M. Fürer, O. Goldreich, Y. Mansour, M. Sipser, and S. Zachos, *On completeness and soundness in interactive proof systems*, in Randomness and Computation, Advances in Computing Research 5, S. Micali, ed., JAI Press, Greenwich, CT, 1989, pp. 429–442.

[14] J. Hartmanis, *Generalized Kolmogorov complexity and the structure of feasible computations*, in Proceedings of the 24th IEEE Symposium on Foundations of Computer Science, Tucson, AZ, 1983, pp. 439–445.

[15] J. Hartmanis and R. Stearns, *On the computational complexity of algorithms*, Trans. Amer. Math. Soc., 117 (1965), pp. 285–306.

[16] F. Hennie and R. Stearns, *Two tape simulation of multitape Turing machines*, J. ACM, 13 (1966), pp. 533–546.

[17] K.-I Ko, *On the complexity of learning minimum time-bounded turing machines*, SIAM J. Comput., 20 (1991), pp. 962–986.

[18] M. Kummer, *On the complexity of random strings (extended abstract)*, in Proceedings of the 13th Annual Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Comput. Sci. 1046, Springer-Verlag, Berlin, 1996, pp. 25–36.

[19] L. Levin, *personal communication*, 1994.

[20] M. Li and P. Vitányi, *An Introduction to Kolmogorov Complexity and Its Applications*, 2nd ed., Grad. Texts Comput. Sci., Springer-Verlag, Berlin, 1997.

[21] T. Long, *Strong nondeterministic polynomial-time reducibilities*, Theoret. Comput. Sci., 21 (1982), pp. 1–25.

[22] C. Lund, L. Fortnow, H. Karloff, and N. Nisan, *Algebraic methods for interactive proof systems*, J. ACM, 39 (1992), pp. 859–868.

[23] D. Martin, *Completeness, the recursion theorem and effectively simple sets*, Proc. Amer. Math. Soc., 17 (1966), pp. 838–842.

[24] A. Shamir, *IP = PSPACE*, J. ACM, 4 (1992), pp. 869–877.

[25] M. Sipser, *A complexity theoretic approach to randomness*, in Proceedings of the 15th ACM Symposium on Theory of Computing, Boston, MA, 1983, pp. 330–335.

[26] L. Valiant and V. Vazirani, *NP is as easy as detecting unique solutions*, Theoret. Comput. Sci., 47 (1986), pp. 85–93.

[27] C. K. Yap, *Some consequences of non-uniform conditions on uniform classes*, Theoret. Comput. Sci., 26 (1983), pp. 287–300.

[28] S. Zachos, *Probabilistic quantifiers and games*, J. Comput. System Sci., 36 (1988), pp. 433–451.

[29] D. Zuckerman, *Randomness-optimal sampling, extractors, and constructive leader election*, in Proceedings of the 28th ACM Symposium on Theory of Computing, Philadelphia, PA, 1996, pp. 286–295.