Randomness is Hard

Harry Buhrman*
CWI
PO Box 94079
1090 GB Amsterdam
The Netherlands

Leen Torenvliet†
University of Amsterdam
Department of Computer Science
Plantage Muidergracht 24
1018 TV Amsterdam
The Netherlands

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 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_s^{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:*

- $MA \subseteq NP^{R_t^{CD}}$, *where MA is the class of Merlin-Arthur games defined by Babai.*
- $AM \subseteq NP^{R_t^{CND}}$, *where AM is the class of Arthur-Merlin games.*
- $PSPACE \subseteq NP^{R_s^{CS}}$.

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.*

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 [BT94].)

Kolmogorov [Lev94] however suggested to focus 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.

In the sixties, when the theory of Kolmogorov complexity was developed, Martin [Mar66] showed that the *co-RE* set of Kolmogorov random strings is complete with respect to (resource unbounded) Turing reductions. Recently Kummer [Kum96] has shown that this can be strengthened to show that this set is also truth-table complete.

The resource bounded version of the random strings was first studied by Ko [Ko91]. The *polynomial* time bounded Kolmogorov complexity $C^p(x)$, for p a polynomial is the smallest program that prints x in $p(|x|)$ steps. 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 *co-NP* under strong nondeterministic polynomial time reductions. He also constructed an oracle where this set is not complete for *co-NP* under deterministic polynomial time Turing reductions.

*Email: buhrman@cwi.nl. Partially supported by the Dutch foundation for scientific research (NWO) by SION project 612-34-002, and by the European Union through NeuroCOLT ESPRIT Working Group Nr. 8556, and HC&M grant nr. ERB4050PL93-0516.

†Research done while on leave at CWI. Email: leen@wins.uva.nl.

Buhrman and Mayordomo [BM95] 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 [Sip83]. 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 $R_t^{CD} = \{x \mid CD^t(x) \geq |x|\}$, for t a fixed polynomial is hard for MA under nondeterministic reductions. MA is the class of Merlin-Arthur games introduced by Babai [Bab85]. As an immediate consequence we obtain that BPP and NP^{BPP} are in $NP^{R_t^{CD}}$.

Next we shift our attention to the nondeterministic distinguishing complexity [BF97], $CND^t(x)$, which is defined as the size of the smallest *nondeterministic* algorithm that runs in time $t(n)$ and accepts only x. We define $R_t^{CND} = \{x : CND^t(x) \geq |x|\}$, for t a fixed polynomial. We show that $AM \subseteq NP^{R_t^{CND}}$ where AM is the class of Arthur-Merlin games [Bab85]. It follows that the complement of the graph isomorphism problem, $\overline{GI}$, is in $NP^{R_t^{CND}}$ and that if for some polynomial t, $R_t^{CND} \in NP \cap co\text{-}NP$ then $GI \in NP \cap co\text{-}NP$.

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 [Har83]. Likewise we define $R_s^{CS} = \{<x,y> : CS^s(x|y) \geq |x|\}$ for $s(n)$ a polynomial. We show that $PSPACE \subseteq NP^{R_s^{CS}}$.

For the first two results we use the oblivious sampler construction of Zuckerman [Zuc96], a Lemma [BF97] 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 [LFKN90, Sha92] for QBF.

Last we construct an oracle world where our first result can not be improved to deterministic reductions. We show that there is an oracle such that $BPP \not\subseteq P^{R_t^{CD}}$ for any polynomial t. The construction of the oracle is an extension of the techniques developed in Beigel et al. [BBF98].

2 Definitions and Notations

We assume the reader familiar with standard notions in complexity theory as can be found e.g., in [BDG88]. 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 [Zuc96].

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.) We define the Kolmogorov complexity function $C(x|y)$ (see [LV97]) by $C(x|y) = \min\{|p| : U(p,y) = x\}$. We define unconditional Kolmogorov complexity by $C(x) = C(x|\lambda)$. Hartmanis defined a time bounded version of Kolmogorov complexity in [Har83], but resource bounded versions of Kolmogorov complexity date back as far as [Bar68]. (See also [LV97].) Sipser [Sip83] defined the distinguishing complexity CD^t. We will need the following versions of resource bounded Kolmogorov complexity and distinguishing complexity.

- $CS^s(x|y) = \min\{|p| : U(p,y) = x$ and $U(p,y)$ uses at most $s(|x|+|y|)$ space $\}$. (See [Har83].)
- $CD^t(x|y) = \min\{p : U(p,x,y)$ accepts and $U(p,z,y)$ rejects for all $z \neq x$ and $U(p,z,y)$ runs in at most $t(|z|+|y|)$ steps for all $z \in \Sigma^*\}$. (See [Sip83].)
- $CND^t(x|y) = \min\{p : U_n(p,x,y)$ accepts and $U_n(p,z,y)$ rejects for all $z \neq x$ and $U_n(p,z,y)$ runs in at most $t(|z|+|y|)$ steps for all $z \in \Sigma^*\}$. (See [BF97].)

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

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

Note that for $\epsilon = 1$ these sets are the sets mentioned in the introduction. We also define the set of strings of maximal space bounded complexity.

$$R^{CS}_s = \{<x, y> : CS^s(x|y) = |x|\}$$

All quantifiers used in this paper are polynomially bounded. Often the particular polynomial is not important for the sequel 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. E.g., 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 $\{0,1\}^{|x|^c}$.

Let $\#M$ represent the number of accepting computations of a nondeterministic Turing machine M. A language L is in $\oplus P$ if there exists a polynomial time 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 only be interested in functions g that are polynomial.

3 Distinguishing Complexity for Derandomization

In this section we prove hardness of R^{CD}_t and R^{CND}_t for Arthur-Merlin and Merlin-Arthur games respectively under NP-reductions.

Theorem 3.1 *For $0 < \epsilon \leq 1$ and any t with $t(n) \in \omega(n \log n)$, $MA \subseteq NP^{R^{CD}_{t,\epsilon}}$*

and

Theorem 3.2 *For $0 < \epsilon \leq 1$ and any t with $t(n) \in \omega(n \log n)$, $AM \subseteq NP^{R^{CND}_{t,\epsilon}}$*

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 can in general not be substituted by computational randomness.

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

Proof . Suppose for a contradiction that for any constant d_0 and infinitely many s with $CD^t(s) \geq g(n)$, it holds that $CD^{t'}(s[1..f(|s|)]) < g(f(|s|)) - 2\log|f(|s|)| - d_0$. 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 $|p_s| < g(f(|s|)) - 2\log|f(|s|)| - d_0$. 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 [HS66].
Its length is $|p_s| + |s| - f(|s|) + \log|f(|s|)| + d_1 < g(f(|s|)) - 2\log|f(|s|)| - d_0 + |s| - f(|s|) + \log|f(|s|)| + d_1$. Which is less than $g(|s|)$ if we take $d_0 > d_1$. Hence $CD^t(s) < g(|s|)$, a contradiction. □

Corollary 3.4 *For every polynomial n^c, $0 < \epsilon \leq 1$, $t \in \omega(n\log n)$ and sufficiently large string s with $CD^t(s) \geq \epsilon|s|$, if $m = |s|^{\frac{1}{c}}$ and $s' = s[1..m]$ then $CD^{n^c}(s') \geq \epsilon|s'| - 2\log|s'| - O(1)$.*

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

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 3.5 ([Zuc96]) *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 3.6 ([BF97]) *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 [BF97], an analogous lemma holds for CND^p and NP. That is:

Lemma 3.7 ([BF97]) *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. First we use Theorem 3.5 to amplify MA and AM protocols using as few extra random bits as possible.

Lemma 3.8

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 from $\{0,1\}^{(1+\alpha)(1+k)m}$

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 from $\{0,1\}^{(1+\alpha)(1+k)m}$

Proof .

1. Zachos and Fürer showed that the fraction 2/3 can be replaced by 1 in [ZF87]. 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 3.5 with $\gamma = 2^{-km}$, and $\epsilon = 1/6$. For given x and y let f_{xy} be the 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 $Ef_{xy} < 1/3$ for all y. With probability $1-\gamma$ the sample points returned by the oblivious sampler are such that $\left|\frac{1}{d}\sum_{i=1}^d f_{xy}(z_i) - Ef_{xy}\right| \leq \epsilon$, so $\frac{1}{d}\sum_{i=1}^d f(z_i) > \frac{1}{2}$ with probability $\leq 2^{-km}$. □

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 [ZF87], but now to Theorem 2(ii) of that paper. 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 NP computable function. The

sample points $z_1, \dots, z_d$ that are returned in this case have the following properties. If $x \in L$ then $f_x(z_i) = 1$. That is for every possible sample point there is a y_i such that $M_L(x, y_i, z_i) = 1$. So for any set of sample points $z_1, \dots, z_d$ that the sampler may return, there exists a $y = <y_1, \dots, y_d>$ such that $M_L(x, y_i, z_i) = 1$ for all 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 $\Pr\left[(\exists y = y_1 \dots y_d)[\frac{1}{d}\sum_{i=1}^{d} M_L(x, y_i, z_i) > \frac{1}{2}]\right]$ is less than 2^{-km}. So if we let $M(x, y, r)$ be the machine that uses r to generate d sample points and then interprets y as $<y_1, \dots, y_d>$ 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* a *MA* (*AM*) protocol.

Lemma 3.9

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

Proof .

1. Let M be the deterministic $q(n)$ time bounded machine corresponding to L of Lemma 3.8, item 1. Choose $\alpha < \frac{\epsilon}{2}$ and $k > \frac{6}{\epsilon - 2\alpha}$. Let r be such that $CD^q(r) \geq \epsilon|r|$.

   Suppose $x \in L$. Then it follows that there exists a y such that for all s, $M(x, y, s) = 1$. So in particular it holds that $M(x, y, r) = 1$.

   Suppose $x \notin L$. We have to show that for all y it is the case that $M(x, y, r) = 0$. 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 $A_{x,y_0} \in P$ 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 $|M|+|x|+|y|$.) 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 3.6 that:

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

   This contradicts the choice of r since:

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

2. Let M be the deterministic $q(n)$ time bounded machine corresponding to L of Lemma 3.8, item 2. Choose $\alpha < \frac{\epsilon}{2}$ and $k > \frac{5}{\epsilon - 2\alpha}$. Let r be 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 $A_x \in NP$ 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 3.7 that:

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

   This contradicts the choice of r since:

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

□

The following corollary shows that a string of high enough CD^{poly} complexity can be used to derandomize a *BPP* machine.

Corollary 3.10 *Let A be a set in BPP. For any $\epsilon > 0$ there exists a polynomial time Turing machine M 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 $q'(n) = (1+\alpha)(1+l)q(n)$ be as in Lemma 3.9, item 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 $s \in R^{CD}_{t,\epsilon}$. Set $r = s[1..q'(n)]$ and accept if and only if there exists a y such that $M(x, y, r) = 1$. By Corollary 3.4 it follows that $CD^q(r) \geq \epsilon|r|$ and the correctness of the reductions follows directly from Lemma 3.9, item 1.

Proof of Theorem 3.2. This follows directly from Lemma 3.9, item 2. The NP-algorithm is analogous to the one above.

Corollary 3.11 *For any $\epsilon > 0$ and $t \in \omega(n \log n)$*

1. *BPP and NP^{BPP} are included in $NP^{R^{CD}_{t,\epsilon}}$.*
2. *$\overline{GI} \in NP^{R^{CND}_{t,\epsilon}}$.*

It follows that if $R^{CND}_{t,\epsilon} \in NP \cap co\text{-}NP$ then the Graph isomorphism problem, GI, is in $NP \cap co\text{-}NP$.

4 Limitations

In the previous section we showed that the set $R^{CD}_{t,\epsilon}$ is hard for MA under NP reductions. One might wonder whether $R^{CD}_{t,\epsilon}$ is also hard for MA under a stronger reduction like the deterministic polynomial time Turing reduction. In this section we show that this, if true, will need a nonrelativizing proof.

To be more specific, we show the existence of an oracle A such that $EXP^{NP^A} \subseteq NP^A/poly$ *and* $P^A = \oplus P^A$. We first show that this implies for $0 < \epsilon \leq 1$ that $BPP^A \not\subseteq P^{R^{CD^A}_{t,\epsilon}}$. (Note that $R^{CD^A}_{t,\epsilon}$ has a meaningful definition.)

Lemma 4.1 *For any oracle A and $0 < \epsilon \leq 1$ it holds that if $EXP^{NP^A} \subseteq NP^A/poly$ and $\oplus P^A = P^A$ then $BPP^A \not\subseteq P^{R^{CD^A}_{t,\epsilon}}$.*

Proof . Suppose for a contradiction that the lemma is not true. If $EXP^{NP} \subseteq NP/poly$ then $EXP \subseteq NP/poly$, so $EXP \subseteq PH$. Also, if $EXP^{NP} \subseteq NP/poly$, then certainly $EXP^{NP} \subseteq EXP/poly$. It then follows from [BH92] that $EXP^{NP} = EXP$, so $EXP^{NP} \subseteq PH$.

If $\oplus P = P$ then unique-SAT (see [BFT97] for a definition) is in P. Then $NP = R$ by [VV86] and hence $PH \subseteq BPP$.

Finally unique-SAT is in P is equivalent to: For all x and y, $C^{poly}(x|y) \leq CD^{poly}(x|y) + O(1)$. (See [FK96].) As R^{C^p} (the set of polynomial time C random strings is in $co\text{-}NP$) it follows from the proof of that theorem that for a particular universal machine[1] unique-SAT $\in P$ implies $R^{CD}_{t,\epsilon} \in co\text{-}NP$. This in its turn implies by assumption that BPP and hence EXP^{NP} are in P^{NP}, this however contradicts the hierarchy theorem for alternating Turing machines [HS65]. As all parts of this proof relativize, we get the result for any oracle. □

Now we proceed to construct the oracle.

Theorem 4.2 *There exists an oracle A such that $EXP^{NP^A} \subset NP^A/poly \wedge \oplus P^A = P^A$*

Proof . The proof of the construction parallels the one from Beigel, Buhrman and Fortnow [BBF98], who construct an oracle such that $P^A = \oplus P^A$ and $NEXP^A = NP^A$. 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 complete set for NP^A. Let N^{K^A} be a deterministic machine that runs in time 2^n and for all A accepts a language H^A that is complete for EXP^{NP^A}. We will construct A such that there exists a n^2 bounded advice function f such that for for all w

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

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

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

First we give some intuition for the proof. 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 $<1, z, w, v>$ as a variable $y_{<z,w,v>}$ whose value determines whether $<1, z, w, v>$ is in A. We will show that the computation $N^{K^A}(x)$ can be *forced* in such a way that it can be represented by a low-degree

[1]Contradicting "For all universal machines, $BPP \in P^{R^{CD}_{t,\epsilon}}$" just requires disproving this for a particular universal machine.

polynomial over these variables in the field of two elements. 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 $z = <1, z, w, v>$ we let p_z be the single variable polynomial $y_{<z,w,v>}$.

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 $z = <0, x, 1^{|x|^2}>$ 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 $y_{<z,w,v>}$ (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 4.3 *For each 0-string $z = <0, x, 1^{|x|^2}>$ 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: Simple proof 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 condition 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 4.4 *The representation of the functions $OR(u_1, \ldots, u_m)$ and the $AND(u_1, \ldots, u_m)$ 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

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

W.l.o.g. we will assume that machine N only queries 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 for all $q \in S$ 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 put 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 4.5 *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 . Consider the computation of machine $M_K^A(q)$ that accepts K^A. Let $o_1, \ldots, o_l$ be the smallest number of strings such that adding them to A is an S-consistent extension of A such that $M_K^{A'}(q)$ accepts. (Recall $A' = A \cup \{o_1, \ldots, o_l\}$.) 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:

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

After adding the strings $o_1, \ldots, o_l$ to A we have that $P_q = 1$. Moreover by Lemma 4.3 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 $o_1, \dots, o_l$ 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 $o_1, \dots, o_l$. Since by Lemma 4.4 the degree of the unique multivariate multilinear polynomial that computes the AND over l variables over GF[2] is l it follows that $l \leq 2^{3n}$. □

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 $\|S\| \leq 2^{2n}$ and that at this point by Lemma 4.5 at most 2^{5n} strings have been added to A at this stage. A standard counting argument shows that there is a string z of length n^2 such that no strings of the form $<1, z, w, v>$ 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 $<1, z, x_i, w>$ 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 $<1, z, x_i, w>$ 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 4.6 *Let $\|S\| \leq 2^{2n}$ 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 $<1, z, x_i, w>$ with $|w| = n^2$.*

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

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

Where P_q is defined as in Lemma 4.5, equation 2. Initially $Q_{x_i} = 0$ and the degree of $Q_{x_i} \leq 2^{5n}$. Since every extension of A with strings of the form $<1, z, x_i, w>$ is not S consistent it follows that Q_{x_i} computes the OR of the variables $y_{<z, x_i, w>}$. Since there are 2^{n^2} many of those variables we have by Lemma 4.4 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 $<1, z, x_i, w>$ and x_i is properly coded into A. □

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

□

Our oracle also extends the oracle of Ko [Ko91] to CD^{poly} complexity as follows.

Corollary 4.7 *There exists an oracle such that $\overline{R_{t,\epsilon}^{CD}}$ for any $t \in \omega(n \log(n))$ and $\epsilon > 0$ is complete for NP under strong nondeterministic reductions and $P^{NP} \neq \Sigma_2^p$*

Proof. The oracle from Theorem 4.2 is a world where *co-NP* $\subseteq$ *BPP* and $C^{poly}(x|y) = CD^{poly}(x|y) + O(1)$, hence it follows that $\overline{R_{t,\epsilon}^{CD}} \in NP$. Moreover Corollary 3.11 relativizes so by Item 1 we have that $BPP \subseteq NP^{\overline{R_{t,\epsilon}^{CD}}}$. □

As a byproduct our oracle shows the following.

Corollary 4.8 *$\exists A$ Unique-SAT$^A \in P^A$ and $P^{NP^A} \neq \Sigma_2^{p,A}$*

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

5 *PSPACE* and R_s^{CS}

In this section we further study the connection between R_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 a constant number of 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 class of quantified boolean formulas (QBF) is defined as the closure of the set of boolean variables x_i and their negations $\overline{x_i}$ under the operations $\wedge$ (and), $\vee$ (or), $\forall x_i$ (universal quantification) and $\exists x_i$ (existential quantification). 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 [Sha92].

Definition 5.1 ([Sha92]) *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 ([Sha92]) *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 ([Sha92]) *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 $\otimes_i = +$ if $Q_i = \exists$ and $\otimes_i = \times$ 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 $\perp(B)$ the arithmetization of B'.

Theorem 5.4 ([Sha92]) *The language of all closed simple true QBFs is complete for PSPACE (under polynomial time many-one reductions).*

Theorem 5.5 ([Sha92]) *A simple closed quantified boolean formula B is true if and only if 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 *any such prime.*

Theorem 5.6 ([Sha92]) *The functional form of every simple QBF can be represented by a univariate polynomial of degree at most* 3.

Theorem 5.7 ([Sha92]) *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 if and only if B is true. Then, B's functional form F (recall: 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 $p_1(0) \otimes p_1(1) = V$. (Here $\otimes$ is $+$ if the first quantifier is $\exists$ and $\times$ if the first quantifier is $\forall$.) 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 $p_n(r_1, \ldots, r_n) = \perp(B)(r_1, \ldots, r_n)$. Two more things are needed: First, if any *other* sequence of polynomials $q_1, \ldots, q_n$ has the property that $q_1(0) \oplus q_1(1) \neq V$ and $p_n(r_1, \ldots, r_n) = \perp(B)(r_1, \ldots, r_n)$, then there has to be some i where $q_i(r_i) = p_i(r_i)$, yet $q_i \neq p_i$. I.e., 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 ([Sha92]) *Let B be a closed simple QBF wherein the quantors are $Q_1, \ldots Q_n$ 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 $r_1, \ldots, r_n$ 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. *$p_1(0) \otimes_1 p_1(1) = V$ and $p_1(0) \otimes_1 p_1(1) > 0$ iff B is true.*
2. *$p_{i+1}(0) \otimes_{i+1} p_{i+1}(1) = p_i(r_i)$*
3. *$p_n(r_n) = \perp(B)(r_1, \ldots, r_n)$*
4. *For any sequence of univariate polynomials $q_1, \ldots, q_n$ such that:*

(a) $p_1(0) \otimes_1 p_1(1) \neq q_1(0) \otimes_1 q_1(1)$ and

(b) $q_{i+1}(0) \otimes_{i+1} q_{i+1}(1) = q_i(r_i)$ and

(c) $q_n(r_n) = \perp(B)(r_1, \dots, r_n)$

there is a minimal i such that $p_i \neq q_i$, yet $p_i(r_i) = q_i(r_i)$. I.e., 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, $r_1, \dots, r_{i-1}$.*

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 the value that can be computed from q_1 in that sequence 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 the 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 *Let B be a simple closed QBF on n variables. Let P be a prime number of size polynomial in $|B|$. Let $q_1 \dots q_n$ be a sequence of polynomials of degree 3 with coefficients in $[1..P]$. and let $r_1, \dots, r_n$ be numbers in $[1..P]$ such that $CS^n(r_i \mid <B, P, q_1, \dots, q_i, r_1, \dots, r_{i-1}, 0^{(|B|+|P|)^2}>) \geq |P|$.*

Furthermore suppose that for $i \geq 2$, $q_{i-1}(r_{i-1}) = q_i(0) \otimes_i q_i(1)$. If B is false and $\perp(B)(r_1, \dots, r_n) = q_n(r_n)$ then $q_1(0) \otimes_1 q_1(1) = 0$, when these equalities are taken modulo P.

Proof: Take all calculations modulo P. Suppose $q_1(0) \otimes_1 q_1(1) \neq 0$. 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 $p_1(0) \otimes_1 p_1(1) = 0$ modulo *any* prime, so $p_1(0) \otimes_1 p_1(1) \neq q_1(0) \otimes_1 q_1(1)$. 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 $CS^n(r_i \mid <B, P, q_1, \dots, q_i, r_1, \dots, r_{i-1}, 0^{(|B|+|P|)^2}>)$ is bounded by a constant. A contradiction. □

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

Theorem 5.10 *$PSPACE \subseteq NP^{R_s^{CS}}$*

Proof: We prove the lemma for $s(n) = n$, 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 quantified boolean formulas as follows. On input B first check that B is simple. Guess a prime number P of size polynomial in B, a sequence of polynomials $p_1, \dots, p_n$ of degree at most 3 and with coefficients in $[1..P]$. Finally guess a sequence of numbers $r_1, \dots, r_n$. Check that:

1. $p_1(0) \otimes_1 p_1(1) > 0$ and
2. $p_{i+1}(0) \otimes_{i+1} p_{i+1}(1) = p_i(r_i)$ and
3. $p_n(r_n) = \perp(B)(r_1, \dots, r_n)$ and
4. finally that $CS^n(r_i | <B, P, p_1, \dots, p_i, r_1, \dots, r_{i-1}, 0^{(|B|+|P|)^2}>)$ is at least $|P|$ for all $i \leq n$.

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 $p_1(0) \otimes_1 p_1(1) = 0$, so the first check must fail. □

By the fact that *PSPACE* is closed under complement and the fact that R_s^{CS} is also in *PSPACE* Theorem 5.10 gives that R_s^{CS} is complete for *PSPACE* under strong nondeterministic reductions [Lon82].

Corollary 5.11 *R_s^{CS} is complete for PSPACE under strong nondeterministic reductions*

Buhrman and Mayordomo [BM95] 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 [BM95] one can prove that the set $RD_t^C = \{<x, y> : C^t(x|y) \geq |x|\}$ is not hard for *EXP* under Turing reductions. On the other hand the proof of Theorem 5.10 also works for this set: $PSPACE \subseteq NP^{RD_t^C}$. We suspect that it is possible to extend this to show that $EXP \subseteq NP^{RD_t^C}$. So far, we have been unable to prove this.

Acknowledgements

We thank Paul Vitányi for interesting discussions and providing the title of this paper.

References

[Bab85] L. Babai. Trading group theory for randomness. In *Proc. 3rd ACM Symp. Theory of Computing*, pages 421–429, 1985.

[Bar68] Ja. M. Barzdin. Complexity of programs to determine whether natural numbers not greater than n belong to a recursively enumerable set. *Soviet Math. Dokl.*, 9:1251–1254, 1968.

[BBF98] R. Beigel, H. Buhrman, and L. Fortnow. NP might not be as easy as detecting unique solutions. In *Thirtieth Annual ACM Symposium on Theory of Computing (STOC)*, 1998. To appear.

[BDG88] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I.* Springer-Verlag, 1988.

[BF97] H. Buhrman and L. Fortnow. Resource bounded kolmogorov complexity revisited. In Reischuk and Morvan, editors, *14th Annual Symposium on Theoretical Computer Science*, volume 1200 of *Lecture Notes in Computer Science*, pages 105–116. Springer, 1997.

[BFT97] H. Buhrman, L. Fortnow, and L. Torenvliet. Six hypotheses in search of a theorem. In *Proceedings 12th annual IEEE Conference on Computational Complexity*, pages 2–12, Ulm DE, 1997. IEEE Computer Society Press.

[BH92] H. Buhrman and S. Homer. Superpolynomial circuits, almost sparse oracles and the exponential hierarchy. In R. Shyamasundar, editor, *Proc. 12th Conference on the Foundations of Software Technology & Theoretical Computerscience, Lecture Notes in Computer Science*, pages 116–127. Springer Verlag, 1992.

[BM95] H. Buhrman and E. Mayordomo. An excursion to the kolmogorov random strings. In *Proceedings Structure in Complexity Theory, 10^{th} annual conference (STRUCTURES95)*, pages 197 - 205, Minneapolis, 1995. IEEE Computer Society Press.

[BT94] H. Buhrman and L. Torenvliet. On the structure of complete sets. In *Proc. Structure in Complexity Theory 9th annual conference*, pages 118–133, Amsterdam, Holland, 1994. IEEE computer society press.

[FK96] L. Fortnow and M. Kummer. Resource-bounded instance complexity. *Theoretical Computer Science A*, 161:123–140, 1996.

[Har83] J. Hartmanis. Generalized Kolmogorov complexity and the structure of feasible computations. In *Proc. 24th IEEE Symposium on Foundations of Computer Science*, pages 439–445, 1983.

[HS65] J. Hartmanis and R. Stearns. On the computational complexity of algorithms. *Trans. Amer. Math. Soc.*, 117:285–306, 1965.

[HS66] F. Hennie and R. Stearns. Two tape simulation of multitape Turing machines. *J. Assoc. Comput. Mach.*, 13(4):533–546, 1966.

[Ko91] K. Ko. On the complexity of learning minimum time-bounded turing machines. *SIAM J. Comput.*, 20:962–986, 1991.

[Kum96] Martin Kummer. On the complexity of random strings (extended abstract). In *13th Annual Symposium on Theoretical Aspects of Computer Science*, volume 1046 of *Lecture Notes in Computer Science*, pages 25–36, Grenoble, France, 22–24 February 1996. Springer.

[Lev94] L. Levin. Personal communication. 1994.

[LFKN90] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. In *Proc. 31st Symposium on Foundations of Computer Science*, pages 2–90, New York, 1990. IEEE.

[Lon82] T. Long. Strong nondeterministic polynomial-time reducibilities. *Theoretical Computer Science*, 21:1–25, 1982.

[LV97] Ming Li and P.M.B. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications.* Graduate Texts in Computer Science. Springer-Verlag, second edition, 1997.

[Mar66] D.A. Martin. Completeness, the recursion theorem and effectively simple sets. *Proc. Am. Math. Soc.*, 17:838–842, 1966.

[Sha92] A. Shamir. IP=PSPACE. *Journal of the ACM*, 4:869–877, October 1992.

[Sip83] M. Sipser. A complexity theoretic approach to randomness. In *Proc. 15th ACM Symposium on Theory of Computing*, pages 330–335, 1983.

[VV86] L. Valiant and V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, 1986.

[ZF87] Zachos and Furer. Probabilistic quantifiers vs. distrustful adversaries. *Foundations of Software Technology and Theoretical Computer Science*, 7, 1987.

[Zuc96] D. Zuckerman. Randomness-optimal sampling, extractors, and constructive leader election. In *Proceedings of the 28th ACM Symposium on Theory of Computing*, pages 286–295, 1996.