Electronic Colloquium on Computational Complexity, Report No. 28 (2012)

Kolmogorov Complexity, Circuits, and the Strength of Formal Theories of Arithmetic

Eric Allender*
Rutgers University
allender@cs.rutgers.edu

George Davie
University of South Africa
davieg@unisa.ac.za

Luke Friedman*†
Rutgers University
lbfried@cs.rutgers.edu

Samuel B. Hopkins‡
University of Washington
samhop@uw.edu

Iddo Tzameret§
Tsinghua University
tzameret@tsinghua.edu.cn

Abstract

Can complexity classes be characterized in terms of efficient reducibility to the (undecidable) set of Kolmogorov-random strings? Although this might seem improbable, a series of papers has recently provided evidence that this may be the case. In particular, it is known that there is a class of problems $\mathcal{C}$ defined in terms of polynomial-time truth-table reducibility to R_K (the set of Kolmogorov-random strings) that lies between BPP and PSPACE [4, 3]. In this paper, we investigate improving this upper bound from PSPACE to PSPACE $\cap$ P/poly.

More precisely, we present a collection of true statements in the language of arithmetic, (each provable in ZF) and show that if these statements can be proved in certain extensions of Peano arithmetic, then

$$\text{BPP} \subseteq \mathcal{C} \subseteq \text{PSPACE} \cap \text{P/poly}.$$

We conjecture that $\mathcal{C}$ is equal to P, and discuss the possibility this might be an avenue for trying to prove the equality of BPP and P.

*Supported in part by NSF Grants CCF-0830133, CCF-0832787, and CCF-1064785.

†Supported in part by the [European Community's] Seventh Framework Programme [FP7/2007-2013] under grant agreement n° 238381.

‡Supported in part NSF Grant CCF-1004956 with the DIMACS REU Program.

§Supported in part by the National Basic Research Program of China Grant 2011CBA00300, 2011CBA00301, the National Natural Science Foundation of China Grant 61033001, 61061130540, 61073174

ISSN 1433-8092

1 Introduction

Kolmogorov complexity provides a mathematically precise definition of the set R of "random" strings. Actually, it provides at least *two* distinct but closely-related notions of randomness that we will need to discuss here, one defined in terms of the prefix Kolmogorov complexity function K, and one defined in terms of the plain Kolmogorov complexity function C. This yields the two sets that lie at the center of this paper: $R_K = \{x : K(x) \geq |x|\}$ and $R_C = \{x : C(x) \geq |x|\}$. When it is not important to distinguish between K and C we will simply refer to R.

It is known that PSPACE $\subseteq$ P^R [2], but it is unknown if any larger class such as EXP is in P^R. In this paper we will focus especially on polynomial-time *truth-table* reductions (also known as *non-adaptive* reductions) $\leq^{\mathrm{p}}_{tt}$; our motivation comes in part from a theorem of Buhrman et al., showing BPP $\subseteq \{A : A \leq^{\mathrm{p}}_{tt} R\}$ [4].

Because no larger complexity classes have been shown to be reducible to R in this way, we are interested in the question of whether these inclusions are *optimal* in some sense. It was observed earlier [1] that the class $\{A : A \leq^{\mathrm{p}}_{tt} R_C\}$ contains arbitrarily complex decidable sets (and thus does not look very much like a complexity class), but the same paper also suggested that a more promising avenue was to investigate the classes of problems that are *always* reducible to R, no matter which universal Turing machine was used to define the Kolmogorov functions C and K. This gives rise to the following classes:

Definition 1 *As usual, let Δ^0_1 denote the class of decidable sets. Let C_U denote the plain Kolmogorov complexity function as given by* universal *Turing machine U, and let K_U denote the prefix complexity function as given by universal prefix Turing machine U. Define*

- $\mathcal{C}_C = \Delta^0_1 \cap \bigcap_U \{A : A \leq^{\mathrm{p}}_{tt} R_{C_U}\}$.
- $\mathcal{C}_K = \Delta^0_1 \cap \bigcap_U \{A : A \leq^{\mathrm{p}}_{tt} R_{K_U}\}$.

In each case, the intersection is taken over all *universal* Turing machines U. See Section 2 for more background and definitions relating to Kolmogorov complexity.

The first *upper bounds* on the complexity of sets in $\mathcal{C}_K$ was provided recently: $\mathcal{C}_K \subseteq$ PSPACE [3]. (We conjecture that similar bounds hold for $\mathcal{C}_C$, but at present it is still unknown whether $\mathcal{C}_C = \Delta^0_1$.) Thus, in particular, we have

$$\text{BPP} \subseteq \mathcal{C}_K \subseteq \text{PSPACE} \subseteq \text{P}^R.$$

In this paper, we focus on the following conjecture (which we believe holds for both notions of Kolmogorov complexity, and for all universal machines U):

Conjecture 2 $\mathcal{A} = \{A \in \Delta^0_1 : A \leq^{\mathrm{p}}_{tt} R\} \subseteq$ P/poly.

Our main technical contribution is to present theorems that, in our opinion, support this conjecture. Namely, we build a set of formulas $\{\Psi_A(n,j,k)\}_{A \in \mathcal{A}}$ in the language of Peano Arithmetic, and for each $A \in \mathcal{A}$ present a proof (which can be formalized in certain extensions of Zermelo-Frankel, or ZF) of the statement $\forall n \forall j \forall k \Psi_A(n,j,k)$. We then show that if for each $A \in \mathcal{A}$, and each fixed tuple (**n**,**j**,**k**), the true statement Ψ_A(**n**,**j**,**k**) is provable in certain extensions of Peano Arithmetic, then Conjecture 2 holds. We believe that it is at least plausible that the statements Ψ_A(**n**,**j**,**k**) are, in fact, provable in these extensions of Peano Arithmetic, but we have less confidence in this than in the truth of Conjecture 2. [1]

Note that it is still unknown whether the halting problem is $\leq^{\mathrm{p}}_{tt}$-reducible to R_C (in which case it would hold that $\mathcal{C}_C = \Delta^0_1$). As a consequence of our main result, presenting such a reduction (or presenting a reduction from *any* set outside of P/poly) entails proving independence results from Peano Arithmetic.

Note that, if Conjecture 2 holds, then BPP $\subseteq \mathcal{C}_K \subseteq$ PSPACE $\cap$ P/poly. Thus we think that it is very reasonable to conjecture that $\mathcal{C}_K =$ BPP. But in fact we conjecture more. We believe that $\mathcal{C}_C = \mathcal{C}_K =$ P. In fact, for limited classes of truth-table reductions, equalities of this form are known. In particular, it has been shown that $\Delta^0_1 \cap \bigcap_U \{A : A \leq^{\mathrm{p}}_{dtt} R_{C_U}\} = \Delta^0_1 \cap \bigcap_U \{A : A \leq^{\mathrm{p}}_{\oplus tt} R_{C_U}\} =$ P [1].

In Section 7 we speculate about the possible advantages of pursuing this avenue toward the goal of proving BPP $=$ P. At a minimum, we believe that our results raise the possibility that various mathematical techniques (e.g., from proof theory) might be relevant to the BPP vs. P problem, where such a connection may have seemed less likely before. Certainly the connection surprised some of the authors.

2 A Warm-Up Result

In this section we start with some basic definitions, and then present an easy theorem that provides intuition for Conjecture 2 and whose proof will help motivate some additional definitions.

We say that a language A *polynomial-time truth-table* reduces to a language B, denoted by $A \leq^{\mathrm{p}}_{tt} B$, if there exists a polynomial-time machine M that computes A when given B as an oracle, with the additional requirement that, on input x, M

[1] Recent unpublished work by Burhman and Loff [5] implies that these statements in their current form are in fact independent from the relevant extensions of Peano Arithmetic. Nonetheless, we still believe Conjecture 2 to be true, and we find the connection between these unusual complexity classes and mathematical logic to be of independent interest. See Section 6 for a more in-depth discussion of these developments.

must compute the query set $Q(x)$ of all queries it will ask the oracle B *before* receiving answers to any of its queries.

We will consider only truth-table reductions in this paper; as such we will write M^A to indicate that machine M is using a set A as an oracle, and it will be implicit that the oracle access is non-adaptive.

The plain Kolmogorov complexity of a string x with respect to a Turing machine M is defined as $C_M(x) \doteq \min\{|y| : M(y) = x\}$. A universal Turing machine is a machine U such that for all M and all x, $C_U(x) \leq C_M(x) + c_M$, where c_M is a constant depending only on M. At times the choice of reference machine is not important as long as we choose a universal machine; when this is the case we fix some universal machine U and write $C(x)$ in place of $C_U(x)$. We then define the Kolmogorov random strings to be the set $R_C = \{x : C(x) \geq |x|\}$.

In many settings where Kolmogorov complexity arises, it is more appropriate to use what is known as *prefix* complexity. A Turing machine M is called a *prefix* machine, if, for any string x on which M halts, it is the case that M does not halt on any string of the form xy for any non-empty string y. That is, the domain of the machine must form a prefix code. Given such a prefix machine M, we define $K_M(x) \doteq \min\{|y| : M(y) = x\}$. A universal prefix Turing machine is a prefix machine U such that for all prefix machines M and all x, $K_U(x) \leq K_M(x) + c_M$, where c_M is a constant depending only on M. Similar to the case with plain complexity, we fix some universal prefix machine U and write $K(x)$ in place of $K_U(x)$. We refer to the set of random strings under this version of Kolmogorov complexity as R_K.

All our theorems about random strings from this paper work for both R_C and R_K; we will prove them with respect to R_C and simply write R for the set of random strings, but in Section 5 we indicate how to adjust the proofs to work for R_K as well.

For a set S of binary strings, let $S^{\leq k}$ be the set of all strings in S that have length at most k; i.e $S^{\leq k} = \cup_{i \leq k} S \cap \{0,1\}^i$. Let $\mathcal{V}_k$ be the set of all sets of binary strings that only contain strings of length at most k; i.e $\mathcal{V}_k = \mathcal{P}(\{0,1\}^{\leq k})$, where $\mathcal{P}$ denotes the powerset operation and $\{0,1\}^{\leq k}$ is shorthand for $(\{0,1\}^*)^{\leq k}$.

The complement of R is computably-enumerable; therefore there is a Turing machine E that outputs an enumeration $x_1, x_2, x_3, \ldots$ of all nonrandom strings. We define $R_{k,0} = \{0,1\}^{\leq k}$, and $R_{k,i}$ to be $R_{k,i-1} \setminus \{x_j\}$, where x_j is the ith nonrandom string of length at most k in the enumeration. One can view $R_{k,i}$ as an updated approximation to $R^{\leq k}$ after i nonrandom strings of length at most k have been discovered. Note that for some i^*, $R_{k,i^*} = R^{\leq k}$, and that for all $i > i^*$, $R_{k,i}$ is undefined, since there are no further nonrandom strings of length at most k to be discovered. Even though $R_{k,i}$ is undefined for all $i > i^*$, in order to make the following proposition easier to read we state "$\forall i \exists V \subseteq R_{k,i} \ldots$" as a shorthand for

"for all i for which $R_{k,i}$ is defined, there exists a $V \subseteq R_{k,i} \ldots$" We refer the reader to the Appendix, Section 8 for additional details regarding how to make precise certain details that we present at a more intuitive level.

Proposition 3 *Let $A \in \mathcal{A}$, and let M be a polynomial-time Turing machine running in time $f(n)$ computing a truth-table reduction from A to R. Then*

1. *If $\exists d \forall n \exists V_n \in \mathcal{V}_{d+\log f(n)} \forall x \in \{0,1\}^n \; M^{V_n}(x) = A(x)$, then $A \in$* P/poly.
2. *$\exists d \forall n \forall x \in \{0,1\}^n \forall i \exists V \subseteq R_{d+\log f(n),i}$ such that $M^V(x) = A(x)$.*

This proposition resembles an earlier observation from [1], but adds the condition in Part 2 that $V \subseteq R_{d+\log f(n),i}$. The following is an informal interpretation of the proposition. Part 1 states that if for each n there is some oracle that (a) says that all "long" queries are nonrandom and (b) makes the reduction work for all x of length n, then $A \in$ P/poly. Part 2 says something similar to the hypothesis of Part 1, but weaker: although there might not be a single such oracle that works for all x, for every x there is *some* such oracle that works for that x (and furthermore is a subset of $R^{\leq d+\log f(n)}$). Thus, in some sense it *is* consistent for an oracle to say that all long queries are nonrandom, although this might entail giving incorrect answers to short queries; see Section 6 for more on this topic.

Proof: Part 1 is easy. On inputs of size n the advice string is just an encoding of V_n. Because $|V_n| \leq 2^{d+\log f(n)+1}$, the advice can be encoded using $n^{O(1)}$ bits.

Now, we prove Part 2. Suppose, for the sake of contradiction, that $\forall d \exists n \exists x \in \{0,1\}^n \exists i \forall V \subseteq R_{d+\log f(n),i} \;\; M^V(x) \neq A(x)$.

Let $Q(x')$ be the set of queries that M asks on an input x'. Note that because M runs in time $f(n)$, $|Q(x')| \leq f(|x'|)$. Let T be the Turing machine that, on input (d, r), does a dovetailing search until it finds some tuple (n', x', i') such that for all $V \subseteq R_{d+\log f(n'),i'}$, $M^V(x') \neq A(x')$. (This is where we make use of the assumption that A is decidable.) By our assumptions, it is guaranteed that T will find such a tuple. T then outputs the rth element of $Q(x')$.

The machine T demonstrates that for all queries $z \in Q(x')$, $C(z) \leq 2\log d + \log f(n') + c_T$, where c_T is some constant large enough to encode all the information needed to describe T, including f, E, M and the algorithm N that decides membership in A.

However, for the tuple (n', x', i') that T finds, the oracle $V^* = R^{\leq d+\log f(n')}$ which agrees with R on all short queries and says that all long queries are nonrandom must be bad for x'. That is,

- $V^* = R^{\leq d+\log f(n')} \subseteq R_{d+\log f(n'),i'}$, and
- $M^{V^*}(x') \neq A(x') = M^R(x')$.

Because $M^{V^*}(x') \neq M^R(x')$, there must be some query $z \in Q(x')$ such that $z \in R$ and $|z| > d + \log f(n')$. However, we know that the Kolmogorov complexity of this z is low, so for sufficiently large d this is a contradiction: when d is large enough, we have that $2 \log d + \log f(n') + c_T < d + \log f(n')$.

■

Here is an idea for how we can improve on Proposition 3. The condition that $V \subseteq R_{d+\log f(n),i}$ can be viewed as restricting the set of V's that need to be considered; the proof relies only on the fact that $R^{\leq d+\log f(n)}$ ends up being one of the possible V's. Thus, in the proof of Proposition 3, as the machine T enumerates nonrandom strings as part of its dovetailing search, we can view this process as T "proving" that certain sets V cannot be $R^{\leq d+\log f(n)}$. But enumerating a nonrandom string z such that $z \in V$ is not the *only* way to prove that a set V is not $R^{\leq d+\log f(n)}$. For instance, one can prove that for each k, a constant fraction of strings of length k are in R (see, e.g., [12]). Therefore, if the cardinality of a set V is too small, one can prove that $V \neq R^{\leq d+\log f(n)}$ without explicitly enumerating a nonrandom string z such that $z \in V$. This suggests that we construct the machine T to consider more general proofs that a set V is not equal to $R^{\leq d+\log f(n)}$ than just those proofs based on enumerating nonrandom strings.

This motivates some of the definitions in the next section about formal proof systems.

3 Preliminaries and Notation

3.1 Encoding in Formal Theories

We consider the first-order system Peano Arithmetic (PA) augmented with additional axioms. We will be concerned with languages from the set $\mathcal{A} = \{A \in \Delta_1^0 : A \leq_{tt}^{\mathrm{p}} R\}$. A language $A \in \mathcal{A}$ will be encoded as a finite string $\langle M, N \rangle$, where N is a Turing machine that computes A, and M is a clocked polynomial-time Turing machine computing the truth-table reduction from A to R. Note that any $A \in \mathcal{A}$ can be specified by two such machines; for all $A \in \mathcal{A}$ we fix some such encoding. For a fixed A, we let $t_A(n)$ denote an upper bound on the running time of M, which is bounded by n^c for some constant c.

For a given $A \in \mathcal{A}$ encoded by $\langle M, N \rangle$, PA may not be able to prove that N halts on every input, or that for all x, $M^R(x) = N(x)$. Therefore we define a predicate $\mathrm{Hyp}(A)$, which is an encoding of the sentence "$\forall x$ N halts on input x and $M^R(x) = N(x)$", corresponding to the hypothesis $A \in \mathcal{A}$. For each $A \in \mathcal{A}$, we define the system PA^A to be PA augmented with the additional axiom $\mathrm{Hyp}(A)$. Since $\mathrm{Hyp}(A)$ is true, PA^A is consistent if PA is.

We also define hierarchies based on these PA^A systems as follows. We define PA$_0^A$ to be PA^A, and for each $k > 0$, PA$_k^A$ to be PA$_{k-1}^A$ augmented with an extra axiom con(PA$_{k-1}^A$) stating that PA$_{k-1}^A$ is consistent. We also define PA$_\omega^A$ to be PA^A augmented with an extra axiom encoding "For all k, con(PA$_k^A$)".

The statement of Part 2 of Proposition 3 says "$\exists d \ldots$ " – but in fact we will find it useful to be much more explicit about the value of d. The analysis of Part 2 of Proposition 3 works as long as we pick d so that $c_T + 2\log d < d$. In subsequent arguments we will use a similar style of reasoning, using slightly more complicated machines T, and of course the choice of universal Turing machine U that is used to define Kolmogorov complexity will also contribute, but in all cases $2|\langle M, N\rangle| + 2|U| + 2^{25}$ is a conservative over-estimate on the size of c_T. Thus, if we define d_A to be $8(|\langle M, N\rangle| + |U| + 2^{25})$, and we define $g_A(n)$ to be $d_A + \log t_A(n)$, then we can restate Part 2 of Proposition 3 as follows:

For all $A \in \mathcal{A}$,
$\forall n \forall x \in \{0,1\}^n \forall i \exists V \subseteq R_{g_A(n),i}$ such that $M^V(x) = A(x)$.

Note that the proposition remains true, even if we replace g_A by a somewhat *larger* function. For technical reasons, we will find it useful to define $g_A(n)$ to be $d_A + 2\log n + \log t_A(n)$.

3.2 Other definitions

For a set V we define $L_A(n, V) \doteq \{x \in \{0,1\}^n : M^V(x) = N(x)\}$, where A is encoded as $\langle M, N\rangle$ as described in the previous section. That is, $L_A(n, V)$ is the set of all x's of length n for which M computes the correct answer when V is substituted in as the oracle in the truth-table reduction in place of R.

Later on, we will consider a graph whose vertices correspond to different possible V's, and where a vertex V has "label" $L_A(n, V)$. Recalling the definition of $g_A(n)$ at the end of Section 3.1, note that Part 1 of Proposition 3 still holds when restated as follows:

Proposition 4 *For all $A \in \mathcal{A}$, if $\forall n \exists V_n \in \mathcal{V}_{g_A(n)}$ such that $L_A(n, V_n) = \{0,1\}^n$, then $A \in$* P/poly.

Given any $A \in \mathcal{A}$ and any sets $B \subseteq \mathcal{V}_{g_A(n)}$ and $V \in \mathcal{V}_{g_A(n)}$, we define

$$S_A(n, B, V) \doteq \bigcup_{V' \subseteq V \,:\, V' \notin B} L_A(n, V')$$

Informally, we think of B as an excluded set of sets, or "bad" V's. Thus $S_A(n, B, V)$ is the set of all strings x that "label" some subset of V that is not in the set B.

With these definitions in hand, we can now restate Part 2 of Proposition 3 as follows:

For all $A \in \mathcal{A}$, $\forall n \forall i\ S_A(n, \emptyset, R_{g_A(n),i}) = \{0,1\}^n$.

Restating things once more, we obtain the following useful corollary, which we claim is provable in PA^A for all $A \in \mathcal{A}$:

Corollary 5 *If $S_A(n, \emptyset, V) \neq \{0,1\}^n$, then $\forall i\ V \neq R_{g_A(n),i}$.*

One more definition is necessary. We define $B_A(n, j, k)$ to be the set of all $V \in \mathcal{V}_{g_A(n)}$ such that there is a PA_k^A proof of length at most j of the suitably-encoded sentence "$\forall i, V \neq R_{g_A(n),i}$". Think of $B_A(n, j, k)$ as being a set of V's that can be proved to be "bad" (i.e. not equal to $R^{\leq g_A(n)}$) via a PA_k^A proof of length j.

4 Main Results

Our main focus in this paper is Conjecture 2, which we restate below.

Conjecture 6 $\{A \in \Delta_1^0 : A \leq_{tt}^{\mathrm{p}} R\} \subseteq$ P/poly.

Although we do not prove this conjecture, we do make partial progress in this direction by proving theorems supporting the conjecture and relating it to questions about the provability of certain true sentences in formal theories of arithmetic.

Before stating and proving our main theorem, which concerns a hierarchy of proof systems PA_k^A for various k, we state and prove a simpler version that focuses on PA^A and PA_1^A:

Theorem 7 *Let $\Psi_A(n, j)$ be the formula $\forall i\ S_A(n, B_A(n, j, 0), R_{g_A(n),i}) = \{0,1\}^n$.*

1. *For all $A \in \mathcal{A}$, the sentence $\forall n \forall j\ \Psi_A(n, j)$ is true and provable in PA_1^A.*

2. *If for all $A \in \mathcal{A}$, and each fixed pair $(\mathbf{n}, \mathbf{j})$, PA^A proves $\Psi_A(\mathbf{n}, \mathbf{j})$, then Conjecture 6 is true.*

Proof of Part 2:

Let $\Phi_A(n, j, V)$ be the formula "If $S_A(n, B_A(n, j, 0), V) \neq \{0,1\}^n$ then $\forall i\ V \neq R_{g_A(n),i}$". Note that

$$\Psi_A(n, j) \rightarrow \Phi_A(n, j, V) \tag{1}$$

and this implication is provable in PA^A.

Suppose that for each $A \in \mathcal{A}$ and each fixed pair $(\mathbf{n}, \mathbf{j})$, PA^A proves $\Psi_A(\mathbf{n}, \mathbf{j})$.

Let $A \in \mathcal{A}$ and $\langle M, N \rangle$ be the encoding of A. To prove Conjecture 6 we must show that $A \in$ P/poly. Suppose for contradiction that $A \notin$ P/poly. Then, for some n, by Proposition 4 there does not exist a set $V \in \mathcal{V}_{g_A(n)}$ such that $L_A(n, V) = \{0,1\}^n$.

Choose an $\mathbf{n}$ with this property. We define a directed graph $G_\mathbf{n}$ as follows. For each $V \in \mathcal{V}_{g_A(\mathbf{n})}$ there is a node in $G_\mathbf{n}$. The graph $G_\mathbf{n}$ is leveled, with level h containing all V's of cardinality h. There is an edge from a node V to a node V' in $G_\mathbf{n}$ if and only if $V \subset V'$ and $|V'| = |V| + 1$. Thus $G_\mathbf{n}$ is a rooted, layered, directed graph with the empty set as root.

We make use of the following claim:

Claim 8 *For every $V \in \mathcal{V}_{g_A(\mathbf{n})}$ there is a* PA^A *proof of the sentence* $\forall i\ V \neq R_{g_A(\mathbf{n}),i}$.

Proof: The proof is by induction on $|V|$.

For the basis case, when $V = \emptyset$, a simple counting argument that can be formalized in PA^A proves that there are random strings of every length, and hence PA^A proves $\forall i\ \emptyset \neq R_{g_A(\mathbf{n}),i}$.

Now assume inductively that for all $V' \in \mathcal{V}_{g_A(\mathbf{n})}$ such that $|V'| < h$ there is a PA^A proof of the sentence $\forall i\ V' \neq R_{g_\mathbf{A}(\mathbf{n}),i}$. Let $V \in \mathcal{V}_{g_A(\mathbf{n})}$ with $|V| = h$. To prove the claim, it suffices to show that there is a PA^A proof of the sentence $\forall i\ V \neq R_{g_A(\mathbf{n}),i}$.

By the inductive hypothesis, for some $\mathbf{j}'$, we have that $\{V' : V' \in \mathcal{V}_{g_A(\mathbf{n})} \wedge |V'| < h\} \subseteq B_A(\mathbf{n}, \mathbf{j}', 0)$. Since, in the graph $G_\mathbf{n} - B_A(\mathbf{n}, \mathbf{j}', 0)$, V has indegree zero, it follows from the definition of $S_A(\cdot,\cdot,\cdot)$ that PA^A proves

$$S_A(\mathbf{n}, B_A(\mathbf{n}, \mathbf{j}', 0), V) = L_A(\mathbf{n}, V),$$

and by the choice of $\mathbf{n}$ we have $L_A(\mathbf{n}, V) \neq \{0,1\}^\mathbf{n}$. Hence PA proves that $S_A(\mathbf{n}, B_A(\mathbf{n}, \mathbf{j}', 0), V) \neq \{0,1\}^\mathbf{n}$. By assumption we have that PA^A proves $\Psi_A(\mathbf{n}, \mathbf{j}')$, so by (1) we have that PA^A proves "If $S_A(\mathbf{n}, B_A(\mathbf{n}, \mathbf{j}', 0), V) \neq \{0,1\}^\mathbf{n}$ then $\forall i\ V \neq R_{g_A(\mathbf{n}),i}$". Therefore PA^A proves $\forall i\ V \neq R_{g_A(\mathbf{n}),i}$. ∎

Therefore, by Claim 8 we have that PA^A proves $\forall i\ \{0,1\}^{\leq g_A(\mathbf{n})} \neq R_{g_A(\mathbf{n}),i}$. However, by definition, $\{0,1\}^{\leq g_A(\mathbf{n})} = R_{g_A(\mathbf{n}),0}$, which implies that PA^A is inconsistent. By the consistency of PA^A (which is provable in, say, ZF^A), we therefore get a contradiction. Thus we conclude that A is in P/poly. ∎

Proof of Part 1:

Let $A \in \mathcal{A}$ be encoded by $\langle M, N \rangle$, and suppose for contradiction that there exists (n, j) such that $\neg\Psi_A(n, j)$.

This implies that for some i, $S_A(n, B_A(n, j, 0), R_{g_A(n),i}) \neq \{0, 1\}^n$.

Let T be the following machine. On input (n, r), T does a dovetailing search until it finds some tuple (x, j', i') such that x is a string of length n that is not in $S_A(n, B_A(n, j', 0), R_{g_A(n),i'})$. PA^A can argue that under the assumptions, T is guaranteed to find such a tuple. T then computes $Q(x)$, and outputs the rth element of $Q(x)$.

The input (n, r) to T has length at most $2 \log n + \log t_A(n)$. By the discussion at the end of Section 3.1, this implies that for all queries $z \in Q(x)$, $C(z) \leq g_A(n)$, so there can be no $z \in Q(x) \cap R$ such that $|z| > g_A(n)$. Thus PA^A can argue that $M^R(x) = M^{R^{\leq g_A(n)}}(x)$, since these oracles answer all queries of length at most $g_A(n)$ identically, and by the previous sentence they answer queries from $Q(x)$ of length greater than $g_A(n)$ identically as well.

Therefore PA^A can argue the following points:

- $\exists i^* < 2^{g_A(n)+1}\ R^{\leq g_A(n)} = R_{g_A(n),i^*}$.
- $\exists V^* \in \mathcal{V}_{g_A(n)}\ V^* = R_{g_A(n),i^*}$.
- $A(x) = M^R(x) = M^{R^{\leq g_A(n)}}(x) = M^{V^*}(x)$.
- If $V^* \notin B_A(n, j', 0)$ then $x \in S_A(n, B_A(n, j', 0), R_{g_A(n),i'})$.

(The last item follows from the others together with the definition of $S_A(\cdot, \cdot, \cdot)$.)

Therefore, since from the way x was obtained we also have that x is not in the set $S_A(n, B_A(n, j', 0), R_{g_A(n),i'})$, PA^A can conclude that V^* is in $B_A(n, j', 0)$. From the definition of $B_A(n, j', 0)$ this means that PA^A can conclude that there is a length j' proof in PA^A of the sentence $\forall i\ V^* \neq R_{g_A(n),i}$.

However, we have that $V^* = R_{g_A(n),i^*}$, and as the relation $R(k, V, i)$ with intended meaning "$V = R_{k,i}$" can be defined by a Σ^0_1 formula, PA^A can conclude that there is a PA^A proof of $V^* = R_{g_A(n),i^*}$. (See the Appendix, Section 8, for more details on this.) Therefore PA^A proves that PA^A is inconsistent. In PA^A this gets us very little, but in PA$^A_1$ this is a contradiction. Thus PA$^A_1$ proves $\forall n \forall j \Psi_A(n, j)$. ∎

We have been unable to show that the hypothesis for Part 2 of Theorem 7 holds. In fact, there is a reasonable likelihood that the given statements are *not* provable in PA^A, particularly if as in the proof above these statements reduce to PA^A proving its own consistency. (Certainly, the study of Kolmogorov complexity is a rich source of true statements that are not provable [6], and this might merely be yet another manifestation of this phenomenon.)

On the other hand, we suspect that if one has access to *stronger* theories, then it becomes more likely that the required proofs can be carried out. This leads us to our main theorem:

Theorem 9 *Let* $\Psi_A(n,j,k)$ *be the formula* $\forall i\ S_A(n, B_A(n,j,k), R_{g_A(n),i}) = \{0,1\}^n$.

1. *For all* $A \in \mathcal{A}$*, the sentence* $\forall n \forall j \forall k\ \Psi_A(n,j,k)$*) is true and provable in* PA^A_ω.

2. *If for all* $A \in \mathcal{A}$ *and each fixed tuple* $(\mathbf{n},\mathbf{j},\mathbf{k})$ *there exists an* l *such that* PA^A_l *proves* $\Psi_A(\mathbf{n},\mathbf{j},\mathbf{k})$*, then Conjecture 6 is true.*

Proof: The proof of Part 2 is almost identical to that of Theorem 7, and we omit it here.

The proof of Part 1 is very similar to that of Theorem 7 as well, but we include it here for completeness.

Let $A \in \mathcal{A}$ be encoded by $\langle M, N\rangle$ and suppose for contradiction that there exists (n,j,k) such that $\neg\Psi_A(n,j,k)$.

This implies that for some i, $S_A(n, B_A(n,j,k), R_{g_A(n),i}) \neq \{0,1\}^n$.

Let T be the following machine. On input (n,r), T does a dovetailing search, until it finds some tuple (x, j', k', i') such that x is a string of length n that is not in $S_A(n, B_A(n,j',k'), R_{g_A(n),i'})$. PA^A can argue that under the assumptions, T is guaranteed to find such a tuple. T then computes $Q(x)$, and outputs the rth element of $Q(x)$.

The input (n,r) to T has length at most $2\log n + \log t_A(n)$. By the discussion at the end of Section 3.1, this implies that for all queries $z \in Q(x)$, $C(z) \leq g_A(n)$, so there can be no $z \in Q(x) \cap R$ such that $|z| > g_A(n)$. Thus PA^A can argue that $M^R(x) = M^{R^{\leq g_A(n)}}$, since these oracles answer all queries of length at most $g_A(n)$ identically, and by the previous sentence they answer queries from $Q(x)$ of length greater than $g_A(n)$ identically as well.

Thus PA^A can argue the following points:

- $\exists i^* < 2^{g_A(n)+1}\ R^{\leq g_A(n)} = R_{g_A(n),i^*}$.
- $\exists V^* \in \mathcal{V}_{g_A(n)}\ V^* = R_{g_A(n),i^*}$.
- $A(x) = M^R(x) = M^{R^{\leq g_A(n)}}(x) = M^{V^*}(x)$.
- $V^* \notin B_A(n,j',k')$ implies $x \in S_A(n, B_A(n,j',k'), R_{g_A(n),i'})$.

(The last item follows directly from the definition of $S_A(\cdot,\cdot,\cdot)$, along with the preceding items.)

Thus, since from the way x was obtained we also have that x is not in the set $S_A(n, B_A(n, j', k'), R_{g_A(n),i'})$, PA^A can conclude that V^* is in $B_A(n, j', k')$. From the definition of $B_A(n, j', k')$ this means that PA^A can conclude that there is a length j' proof in PA$^A_{k'}$ of the sentence $\forall i\, V^* \neq R_{g_A(n),i}$.

However, we have that $V^* = R_{g_A(n),i^*}$, and as the relation $R(k, V, i)$ with intended meaning "$V = R_{k,i}$" can be defined by a Σ^0_1 formula, PA^A can conclude that there is PA$^A_{k'}$ proof of $V^* = R_{g_A(n),i^*}$. Therefore PA^A proves that PA$^A_{k'}$ is inconsistent. In PA^A_l, for fixed l, there is not much we can conclude from this, since it is not clear how to bound k' by any fixed number. But in PA$^A_\omega$ this is a contradiction. Thus PA$^A_\omega$ proves $\forall n \forall j \forall k \Psi_A(n, j, k)$.

∎

Of course, it would be much more interesting to obtain an unconditional result, proving containment in P/poly, rather than obtaining this inclusion merely on the assumption that these true statements can be proved in one of the given theories. Although it seems plausible[2] that $\Psi_A(\mathbf{n}, \mathbf{j}, \mathbf{k})$ is provable in PA^A_l for some l such that $l > \mathbf{k}$, it is worthwhile considering what a model $\mathcal{M}$ of PA^A_l that does not satisfy $\Psi_A(\mathbf{n}, \mathbf{j}, \mathbf{k})$ would have to look like. In the standard model, the Turing machine T that we construct in the proof of Part 1 will actually never halt (since, in the standard model, the tuple that T "finds" with the needed properties does not exist). Therefore, in $\mathcal{M}$, the "number" t such that T halts after t steps and finds the tuple (x, j', k', i') must be a nonstandard element of the domain. One can easily require that i' be a standard number, but it is not clear to us whether in this type of framework we can force j' and k' to be standard elements. If we could somehow arrange this, then this might be a first step toward proving that the hypothesis of Part 2 holds unconditionally.

The question remains, is there some way to prove that the hypothesis of Part 2 holds that does not mimic our proof of Part 1? Also, we chose to focus on PA in this paper for concreteness and because it is strong enough to formalize the concepts we need. However, to some extent this choice was arbitrary. Is it possible to devise another hierarchy of proof systems based on a system other than PA containing certain properties that would allow us to prove Conjecture 2 using this type of strategy? Or is this type of approach limited in a way that is independent of the particular system that is used?

[2]The comments in this paragraph and the next represent our thoughts at the time when this work was originally submitted for publication. We now know that this "plausible" statement, as currently formulated, is in fact false. See Section 6 for more details.

5 Adapting to Prefix Complexity

The results in the preceding section were proved with respect to the plain Kolmogorov complexity function C. Here, we provide a few comments regarding how to adapt the arguments, so that they carry over to the prefix Kolmogorov complexity function K.

Briefly, the descriptions of the elements y of $Q(x)$ need to be presented as a prefix-free code. The descriptions that we used were of the form (P, n, r) where we can think of P as being a "program", and n and r are numbers. In the analysis for plain complexity, we gave an upper bound on the length of these descriptions, of the form $g_A(n) = d_A + 2\log n + \log t_A(n)$ (and we remarked that the analysis also would carry through if a slightly larger value of $g_A(n)$ were used).

The term "$2\log n$" in this expression comes from the fact that we need to encode the "comma" between n and r in some way, and a very simple way to do this is to simply double each bit of the number n, and then mark the end of "n" with a pair (either 01 or 10) that is *not* doubled.

If we similarly double each bit of r and mark the end of r, then we will obtain a prefix-free encoding scheme, and the analysis will carry through if we just define $g_A(n)$ to be $d_A + 2\log n + 2\log t_A(n)$.

6 Epilogue

Two months after this work was originally submitted for publication, Buhrman and Loff proved some results that bear directly upon our investigation [5]. Buhrman and Loff had read a preliminary version of our paper, and sought to give an unconditional proof of Conjecture 2. Although this conjecture is still open, one of the theorems in [5] can be seen as lending additional support to the conjectured P/poly upper bound on the class of decidable sets polynomial-time truth-table reducible to R. For a polynomial-time reduction from a decidable set A to the undecidable set R, it seems reasonable to hypothesize that the reduction would also work if one used a very high time-complexity approximation to R, such as $R_K^{t(n)}$ for some very rapidly-growing time bound $t(n)$. Buhrman and Loff have shown that, for each decidable set A and polynomial-time truth-table reduction M, it is the case that for *every* large-enough time bound t, if M reduces A to $R_K^{t(n)}$, then $A \in$ P/poly.

In addition, however – the techniques used by Buhrman and Loff also immediately yield that the sentences $\Psi(\mathbf{n},\mathbf{j},\mathbf{k})$ considered here are, in fact, independent of PA$_\ell$. Moreover, they present a polynomial-time reduction M_0 with the property that it can *not* be directly replaced by a reduction that makes queries only of length $O(\log n)$, having as oracle a subset of R. Thus the general approach discussed in here will need to be revised substantially, if it is to be used to obtain a P/poly upper bound on $\{A \in \Delta_1^0 : A \leq_{tt}^{\mathrm{p}} R\}$.

The reduction M_0 alluded to in the preceding paragraph has the property that it obtains no useful information from the oracle. Thus it is still conceivable that one can formulate a notion of "useful" truth-table reductions, for which it still might hold that, for each length n, there is a set of short random strings V that can be used as an oracle to cause the reduction to give the correct answer for all strings of length n. However, it is far from clear how to formulate such a definition.

7 Why Care?

It is popular these days to conjecture that BPP $=$ P, and much of this popularity is owing to results such as those of Impagliazzo and Wigderson [11], who showed that BPP $=$ P if there is a problem in E that requires circuits of exponential size. But note that a proof that BPP $=$ P that proceeds by first proving circuit size lower bounds yields *much* more than "merely" a proof that BPP $=$ P. It also provides a recipe that one can follow, to start with an arbitrary probabilistic algorithm and replace it with an equivalent deterministic one of comparable complexity.

Indeed, Goldreich has recently argued that *any* proof of BPP $=$ P must proceed along these lines, in that any proof that these classes are equal yields pseudorandom generators that are suitable for derandomizing BPP [8, 9].

But there is a catch! Goldreich's proof requires that the BPP $=$ P question be phrased in terms of *promise* problems, rather than using the more traditional definition in terms of language classes, that we have used here.

We do not dispute Goldreich's assertion that the formulation in terms of promise problems is in many ways more natural and useful than the traditional definition. And we certainly agree that it would be much more useful to have a recipe for obtaining derandomizations, rather than merely a proof that a derandomization must exist. But we find it intriguing that a proof that $\mathcal{C}_C =$ P would prove that BPP $=$ P merely by showing that there would be a contradiction otherwise, and owing to the highly non-computable objects in the definition, it is not clear that such a proof would lend itself to an effective construction of a general-purpose derandomization algorithm. (In particular, it is not clear that it would yield the equality of the promise classes.) That is, since such a proof would deliver less than a proof that yields a derandomization, it is at least conceivable that it would be easier to obtain.

We do not wish to suggest that we have any idea of how to obtain such a proof. After all, we are currently unable even to prove $\mathcal{C}_K \subseteq$ P/poly.

Also, it is clear that such a proof must use nonrelativizing techniques. For instance, the work of [4] shows that, for any decidable oracle B, BPP^B is P^B-

truth-table reducible to R_{K_U} for every U. (There is no need to add an oracle to the definition of R_{K_U}.) Thus it is not true that, for every decidable B, $\Delta^0_1 \cap \bigcap_U \{A : A \leq^{\mathrm{p}^B}_{tt} R_{K_U}\} = \mathrm{P}^B$, because Heller [10] has presented such a B relative to which $\mathrm{BPP}^B = \mathrm{NEXP}^B$.

Acknowledgments

We thank Sergei Artemov, Eli Ben-Sasson, Anupam Das, Zeev Dvir, Kaveh Ghasemloo, Russell Impagliazzo, Dieter van Melkebeek, Keng Meng Selwyn Ng, Andre Scedrov, Frank Stephan, Scott Weinstein, Guohua Wu, Yue Yang, and Liang Yu for helpful discussions.

References

[1] E. Allender, H. Buhrman, and M. Koucký. What can be efficiently reduced to the Kolmogorov-random strings? *Annals of Pure and Applied Logic*, 138:2–19, 2006.

[2] E. Allender, H. Buhrman, M. Koucký, D. van Melkebeek, and D. Ronneburger. Power from random strings. *SIAM Journal on Computing*, 35:1467–1493, 2006.

[3] E. Allender, L. Friedman, and W. Gasarch. Limits on the computational power of random strings. In *Proc. of International Conference on Automata, Languages, and Programming (ICALP)*, volume 6755 of *Lecture Notes in Computer Science*, pages 293–304. Springer, 2011. To appear in special issue of Information and Computation for ICALP 2011.

[4] H. Buhrman, L. Fortnow, M. Koucký, and B. Loff. Derandomizing from random strings. In *25th IEEE Conference on Computational Complexity (CCC)*, pages 58–63. IEEE Computer Society Press, 2010.

[5] H. Buhrman and B. Loff. Personal Communication, 2012.

[6] Gregory J. Chaitin. Information-theoretic limitations of formal systems. *Journal of the ACM*, 21(3):403–424, 1974.

[7] J. Girard. *Proof Theory and Logical Complexity*, volume 1. Bibliopolis, Napoli, 1987.

[8] Oded Goldreich. In a world of P=BPP. In *Studies in Complexity and Cryptography*, volume 6650 of *Lecture Notes in Computer Science*, pages 191–232. Springer, 2011.

[9] Oded Goldreich. Two comments on targeted canonical derandomizers. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:47, 2011.

[10] Hans Heller. On relativized exponential and probabilistic complexity classes. *Information and Control*, 71(3):231–243, 1986.

[11] R. Impagliazzo and A. Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In *Proc. ACM Symp. on Theory of Computing (STOC) '97*, pages 220–229, 1997.

[12] M. Kummer. On the complexity of random strings. In *Proc. of Symp. on Theo. Aspects of Comp. Sci. (STACS)*, volume 1046 of *Lecture Notes in Computer Science*, pages 25–36. Springer, 1996.

8 Appendix: Further Encoding Details

Throughout this paper, for the sake of readability, we have presented informally proofs meant to be in formal systems. In this section we attempt to clarify the formalization of a couple key definitions in these proofs.

An important definition, introduced in Section 2, is the definition of the set $R_{k,i}$. Formally, we define $R_{k,i}$ by means of a relation $R(V, k, i)$ that is TRUE if and only if the set V is equal to $R_{k,i}$. (Of course R takes as input an encoding $\langle V \rangle$ of the set V, but we will continue to abuse notation in this way). The quantifier complexity of the formula used to define this relation plays an important role. At the end of the proof of Part 1 of Theorem 7, we state that PA^A proves the implication "$R(V^*, g_A(n), i^*) \to \text{PA}^A \vdash R(V^*, g_A(n), i^*)$" (a similar statement occurs in Theorem 9 as well). Here "PA$^A \vdash R(V^*, g_A(n), i^*)$" is shorthand for a formula encoding that $R(V^*, g_A(n), i^*)$ is provable in PA^A. That this implication involving PA^A actually is provable in PA^A itself depends on $R(V, k, i)$ being definable by a Σ_1^0 formula; i.e., one that can be expressed as $\exists \vec{x}\, R'(\vec{x}, V, k, i)$, where $R'(\vec{x}, V, k, i)$ is a formula containing only bounded quantifiers. (See, for example, [7, Theorems 1.3.4 and 1.4.7] for a proof of this fact.)

Below we show that $R(V, k, i)$ can in fact be defined by a Σ_1^0 formula:

$$\begin{aligned} R(V, k, i) \doteq \exists y\, T(\mathbf{U}, k, i, y) \wedge \exists w \leq y\; out(w, y) \wedge \forall z \in \{0,1\}^{\leq k} \\ z \in V \longleftrightarrow \exists j \leq i\; z = w_j. \end{aligned}$$

Here, $T(\mathbf{U}, k, i, y)$ is a formula expressing that y is the transcript of a halting execution of machine $\mathbf{U}$ on input (k, i), where $\mathbf{U}$ is the Turing machine that takes as input (a, b) and enumerates the first b nonrandom strings of length at most a. (If there do not exist b nonrandom strings of length at most a then no y will satisfy the formula). Also, $out(w, y)$ expresses that w is the output of the execution with transcript y, and w_j stands for the jth element of w (viewing w as a list of strings).

It is standard that the formula $T(\mathbf{U}, k, i, y)$ can be defined by a formula containing only bounded quantifiers.

Note that with the definition $R(V, k, i)$ in hand, we can express a predicate $Z(V, k)$ with intended meaning "$V = R^{\leq k}$" as:

$$Z(V, k) \doteq \exists i^* \leq 2^{k+1}\ R(V, k, i^*) \wedge \forall V' \in \mathcal{V}_k\ \neg R(V', k, i^* + 1).$$

Of course, this predicate $Z(V, k)$ is not Σ_1^0, but it is sufficient for our purposes that $R(V, k, i)$ is.

ECCC ISSN 1433-8092
http://eccc.hpi-web.de