What Can be Efficiently Reduced to the K-random Strings?

Eric Allender*
Rutgers University, New Brunswick, NJ, USA
allender@cs.rutgers.edu

Harry Buhrman
CWI and University of Amsterdam, Amsterdam, Netherlands
buhrman@cwi.nl

Michal Koucký†
McGill University, Montréal, PQ, Canada
mkoucky@cs.mcgill.ca

January 19, 2004

Abstract

We investigate the question of whether one can characterize complexity classes (such as PSPACE or NEXP) in terms of efficient reducibility to the set of Kolmogorov-random strings R_{K}. We show that this question cannot be posed without explicitly dealing with issues raised by the choice of universal machine in the definition of Kolmogorov complexity. Among other results, we show that although for every universal machine U, there are very complex sets that are $\leq^{\mathrm{p}}_{\mathrm{dtt}}$-reducible to R_{K_U}, it is nonetheless true that $\mathrm{P} = \mathrm{REC} \cap \bigcap_U \{A : A \leq^{\mathrm{p}}_{\mathrm{dtt}} R_{\mathrm{K}_U}\}$. We also show for a broad class of reductions that the sets reducible to R_{K} have small circuit complexity.

1 Introduction

The set of *random strings* is one of the most important notions in Kolmogorov complexity theory. A string is random if $\mathrm{K}(x) \geq |x|$. (Given a Turing machine U, $\mathrm{K}_U(x)$ is defined to be the minimum length of a "description" d such that $U(d) = x$. As usual, we fix one such "universal" machine U and define $\mathrm{K}(x)$ to be equal to $\mathrm{K}_U(x)$. In most applications, it does not make much difference which "universal" machine U is picked; it suffices that U satisfies the property that for all U' there exists a constant c such that $\mathrm{K}_U(x) \leq \mathrm{K}_{U'}(x) + c$.) Let R_{K} denote the set of random strings, and let R_{K_U} denote the

*Partially supported by NSF grant CCR-0104823.
†Partially supported by NSF grant CCR-0104823. Part of this work was done while visiting CWI, Amsterdam and while a graduate student at Rutgers University, NJ.

corresponding set when we need to be specific about the particular choice of machine U.

It has been known since [Pos44] that R_{K} is co-r.e. and is complete under weak-truth-table reductions. This was improved significantly by Kummer, who showed that R_{K} is complete under truth-table reductions [Kum96] (even under disjunctive truth-table reductions (dtt-reductions)). Thus there is a computable time bound t and a function f computable in time t such that, for every x, $f(x)$ is a list of strings with the property that $f(x)$ contains an element of R_{K} if and only if x is not in the halting problem. Kummer's argument in [Kum96] is not very specific about the time bound t. Can this reduction be performed in exponential time? Or in doubly-exponential time?[1] In this paper, we provide an answer to this question; surprisingly, it is neither "yes" nor "no".

Kummer's theorem is not primarily a theorem about complexity, but about computability. More recently, however, attention was drawn to the question of what can be *efficiently* reduced to R_{K}. Using derandomization techniques, it was shown in [ABK$^+$02] that every r.e. set is reducible to R_{K} via reductions computable by polynomial-size *circuits*. This leads to the question of what can be reduced to R_{K} by polynomial-time *machines*. In partial answer to this question, it was also shown in [ABK$^+$02] that PSPACE is contained in $\mathrm{P}^{R_{\mathrm{K}}}$.

Question: Is it possible to *characterize* PSPACE in terms of efficient reductions to R_{K}?

Our goal throughout this paper is to try to answer this question. We present a concrete hypothesis later in the paper. Before presenting the hypothesis, however, it is useful to present some of our work that relates to Kummer's theorem, because it highlights the importance of being very precise about what we mean by "the Kolmogorov random strings".

Our first theorem suggests that Kummer's reduction might be computable in doubly-exponential time.

Theorem 1 *There exists a universal Turing machine U such that $\{0^{2^x} : x$ is not in the Halting problem$\}$ is polynomial-time reducible to R_{K_U} (and in fact this reduction is even a $\leq^{\mathrm{p}}_{\mathrm{dtt}}$ reduction).*

Note that, except for the dependence on the choice of universal machine U, this is a considerable strengthening of the result of [Kum96], since it yields a polynomial-time reduction (starting with a very sparse encoding of the halting problem). In addition, the proof is much simpler.

[1] Kummer does show in [Kum96] that completeness under truth-table reductions does *not* hold under some choices of numberings of the r.e. sets; however his results *do* hold for every choice of a universal Turing machine (i.e., "Kolmogorov" numberings, or "optimal Gödelnumberings"). Kummer's result holds even under a larger class of numberings known as "optimal numberings". For background, see [Sch74].

However, the preceding theorem is unsatisfying in many respects. The most annoying aspect of this result is that it relies on the construction of a fairly "weird" universal Turing machine U. Is this necessary, or does it hold for *every* universal machine? Note that one of the strengths of Kolmogorov complexity theory has always been that the theory is essentially insensitive to the particular choice of universal machine. We show that for this question (as well as for other questions regarding the set of Kolmogorov-random strings) the choice of universal machine *does* matter.

1.1 Universal Machines Matter

To illustrate how the choice of universal machine matters, let us present a corollary of our Theorem 12.

Corollary 2 *Let t be any computable time bound. There exists a universal Turing machine U and a decidable set A such that A is not dtt reducible to R_{K_U} in time t.*

Thus, in particular, the reason why Kummer was not specific about the running time of his truth-table reduction in [Kum96] is that no such time bound can be stated, without being specific about the choice of universal Turing machine. This stands in stark contrast to the result of [ABK+02], showing that the halting problem is P/poly-reducible to R_{K}; the size of that reduction does not depend on the universal Turing machine that is used to define R_{K}.

Most notions in complexity theory (and even in computability theory) are invariant under polynomial-time isomorphisms. For instance, using the techniques of [BH77] it is easy to show that for any reasonable universal Turing machines U_1 and U_2, the corresponding halting problems $H_i = \{(x, y) : U_i(x, y) \text{ halts}\}$ are p-isomorphic. However, it follows immediately from Corollary 2 and Theorem 1 that the corresponding sets of random strings $R_{\mathrm{K}_{U_i}}$ are not all p-isomorphic.

Corollary 3 *Let t be any computable time bound. There exist universal Turing machines U_1 and U_2 such that $R_{\mathrm{K}_{U_1}}$ is not reducible to $R_{\mathrm{K}_{U_2}}$ in time t. In particular, $R_{\mathrm{K}_{U_1}}$ and $R_{\mathrm{K}_{U_2}}$ are not isomorphic via isomorphisms computable in time t.*

(We believe that the situation is actually even worse than this, in that the quantifiers in the preceding corollary can be switched. Even if we take U_1 to be the "standard" universal machine, and we define $U_2(0d) = U_1(d)$, we do not see how to construct a computable isomorphism between $R_{\mathrm{K}_{U_1}}$ and $R_{\mathrm{K}_{U_2}}$.)

The lesson we bring away from the preceding discussion is that the choice of universal machine is important, in any investigation of the question of what can be efficiently reduced to the random strings. In contrast, all of the results of [ABK+02] (showing hardness of R_{K}) hold *no matter* which universal Turing machine is used to define Kolmogorov complexity.

Another obstacle that seems to block the way to any straightforward characterization of complexity classes in terms of R_{K} is the fact that, for *every* universal Turing machine and *every* computable time bound t, there is a recursive set A such that $A \leq^{\mathrm{p}}_{\mathrm{dtt}} R_{\mathrm{K}_U}$ but such that $A \notin \mathrm{DSPACE}(t)$ (Theorem 14). Thus $\mathrm{P}^{R_{\mathrm{K}}}$ may not correspond to any reasonable complexity class. How can we proceed from here?

We offer the following hypothesis, as a way of "factoring out" the effects of pathological machines. In essence, we are asking what can be reduced to the K-random strings, *regardless* of the universal machine that is used.

Hypothesis 4 $\mathrm{PSPACE} = \mathrm{REC} \cap \bigcap_U \mathrm{P}^{R_{\mathrm{K}_U}}$.

We are unable to establish this hypothesis (and indeed, we stop short from calling it a "conjecture"). However, we do prove an analogous statement for polynomial-time dtt reductions.

Motivation for studying dtt reductions comes from Kummer's paper [Kum96] (presenting a dtt reduction from the complement of the halting problem to R_{K}), as well as from Theorem 1 and Corollary 2. The following theorem is similar in structure to Hypothesis 4, indicating that it is possible to "factor out" the choice of universal machine in some instances.

Theorem 5 $\mathrm{P} = \mathrm{REC} \cap \bigcap_U \{A : A \leq^{\mathrm{p}}_{\mathrm{dtt}} R_{\mathrm{K}_U}\}$.

We take this as weak evidence that something similar to Hypothesis 4 might be true, in the sense that it shows that "factoring out" the effects of universal machines can lead to characterizations of complexity classes in terms of reducibility to the random strings.

1.2 Approaching the Hypothesis

In order to prove Hypothesis 4, one must be able to show that there are decidable sets that cannot be reduced efficiently to R_{K_U} for some U. Currently we are able to do this only for some restricted classes of polynomial-time truth-table reductions: (a) *monotone* truth-table reductions, (b) parity truth-table reductions, and (c) truth-table reductions that ask at most n^{α} queries, for $\alpha < 1$.

In certain instances, we are able to prove a stronger property. In the case of parity truth-table reductions and disjunctive reductions, if there is a reduction computable in time t from A to R_{K_U} for every U, then A can already be computed nearly in time t. That is, for these classes of reducibilities, a reduction to R_{K} that does *not* take specific properties of the universal machine into account is nearly useless. We believe that this is likely to be true for any polynomial-time truth-table reduction. Note that this stands in stark contrast to polynomial-time Turing reducibility, since PSPACE-complete problems are expected to require exponential time, but can be solved in polynomial time with R_{K} as an oracle. An even stronger contrast is provided by NP-Turing reducibilities. The techniques of [ABK+02] can be used to show that $\mathrm{NEXP} \subseteq \mathrm{NP}^{R_{\mathrm{K}}}$; and thus R_{K} provably provides an exponential speed-up in this setting.

2 Preliminaries and Definitions

In this section we present some necessary definitions. Many of our theorems make reference to "universal" Turing machines. Rather than give a formal definition of what a universal Turing machine is, which might require introducing unnecessary complications in our proofs, we will leave the notion of a "universal" Turing machine as an intuitive notion, and instead use the following properties that are widely known to hold for any natural notion of universal Turing machine, and which are also easily seen to hold for the universal Turing machines that we present here:

- For any two universal Turing machines U_1 and U_2, the halting problems for U_1 and U_2 are p-isomorphic. That is, U_1 halts on input x if and only if U_2 halts on input x' (where x' encodes the information (U_1, x) in a straightforward way). This is a length-increasing and invertible reduction; p-isomorphism now follows by [BH77].

- For any two universal Turing machines U_1 and U_2, there exists a constant c such that $\mathrm{K}_{U_1}(x) < \mathrm{K}_{U_2}(x) + c$.

Let U_1 be the "standard" universal Turing machine. If U_2 is any other machine that satisfies the two properties listed above, then we will consider U_2 to be a universal Turing machine. We are confident that our results carry over to other, more stringent definitions of "universal" Turing machine that one might define. This does not seem to us to be an interesting direction to pursue.

We define $R_{\mathrm{K}_U} = \{x \in \{0,1\}^* : \mathrm{K}_U(x) \geq |x|\}$. When we state a result that is independent of a particular choice of a universal Turing machine U we will drop the U in K_U and refer simply to $\mathrm{K}(x)$.

2.1 Reductions

Let $\mathcal{R}$ be a complexity class and A and B be languages. We define the following types of reductions.

- *Many-one reductions.* We say that A *$\mathcal{R}$-many-one reduces* to B ($A \leq_{\mathrm{m}}^{\mathcal{R}} B$) if there is a function $f \in \mathcal{R}$ such that for any $x \in \Sigma^*$, $x \in A$ if and only if $f(x) \in B$.

- *Truth-table reductions.* We say that A *$\mathcal{R}$-truth-table reduces* to B ($A \leq_{\mathrm{tt}}^{\mathcal{R}} B$) if there is a pair of functions q and r, both in $\mathcal{R}$, such that on an input $x \in \Sigma^*$, function q produces a list of queries $q_1, q_2, \ldots, q_m$ so that for $a_1, a_2, \ldots, a_m \in \{0,1\}$ where $a_i = B(q_i)$, it holds that $x \in A$ if and only if $r(\langle x, (q_1, a_1), (q_2, a_2), \cdots, (q_m, a_m)\rangle) = 1$.

  If $r = \wedge_i a_i$, then the reduction is called a *conjunctive truth-table reduction* ($\leq_{\mathrm{ctt}}^{\mathcal{R}}$). If $r = \vee_i a_i$, then the reduction is called a *disjunctive truth-table reduction* ($\leq_{\mathrm{dtt}}^{\mathcal{R}}$). If the function r computes the parity of $a_1, a_2, \ldots, a_m$, then the reduction is called a *parity truth-table reduction* ($\leq_{\oplus\mathrm{tt}}^{\mathcal{R}}$). If the function r is monotone with respect to

$a_1, a_2, \ldots, a_m$ then the reduction is called a *monotone truth-table reduction* ($\leq^{\mathcal{R}}_{\mathrm{mtt}}$). (A function r is monotone with respect to $a_1, \ldots, a_m$, if for any input x, any set of queries $q_1, \ldots, q_m$, and $a_1, \ldots, a_m, a'_1, \ldots, a'_m \in \{0,1\}$, where for all i, $a_i \leq a'_i$, if r accepts $\langle x, (q_1, a_1), (q_2, a_2), \cdots, (q_m, a_m)\rangle$ then it is also the case that r accepts the tuple $\langle x, (q_1, a'_1), (q_2, a'_2), \cdots, (q_m, a'_m)\rangle$) If the number of queries m is bounded by a constant, then the reduction is called a *bounded truth-table reduction* ($\leq^{\mathcal{R}}_{\mathrm{btt}}$). If the number of queries m is bounded by $f(n)$, then the reduction is called a $f(n)$ *truth-table reduction* ($\leq^{\mathcal{R}}_{f(n)-\mathrm{tt}}$).

- *Turing reductions.* We say that A *$\mathcal{R}$-Turing reduces* to B ($A \leq^{\mathcal{R}}_{\mathrm{T}} B$) if there is an oracle Turing machine in class $\mathcal{R}$ that accepts A when given B as an oracle.

3 Inside $\mathrm{P}^{R_{\mathrm{K}}}$

We have two kinds of results to present in this section. First we present several theorems that do not depend on the choice of universal machine. Then we present our results that highlight the effect of choosing certain universal machines.

3.1 Inclusions that Hold for all Universal Machines

The following is a strengthened version of claims that were stated without proof in [ABK+02].

Theorem 6

1. $\{A \in \mathrm{REC} : A \leq^{\mathrm{p}}_{\mathrm{ctt}} R_{\mathrm{K}}\} \subseteq P$.
2. $\{A \in \mathrm{REC} : A \leq^{\mathrm{p}}_{\mathrm{btt}} R_{\mathrm{K}}\} \subseteq P$.
3. $\{A \in \mathrm{REC} : A \leq^{\mathrm{p}}_{\mathrm{mtt}} R_{\mathrm{K}}\} \subseteq \mathrm{P/poly}$.

Proof. In all three arguments we will have a recursive set A that is $\leq^{(q,r)}_{tt}$ reducible to R_{K}, where (q, r) is the pair of polynomial-time-computable functions defining the $\leq^{\mathrm{p}}_{\mathrm{ctt}}$, $\leq^{\mathrm{p}}_{\mathrm{btt}}$ and $\leq^{\mathrm{p}}_{\mathrm{mtt}}$ reductions, respectively. For $x \in \{0,1\}^*$, $Q(x)$ will denote the set of queries produced by q on input x.

1. (q, r) computes a $\leq^{\mathrm{p}}_{\mathrm{ctt}}$ reduction. For any $x \in A$, $Q(x) \subseteq R_{\mathrm{K}}$. Hence, $Q = \bigcup_{x \in A} Q(x)$ is an r.e. subset of R_{K}. Since R_{K} is immune (i.e., has no infinite r.e. subset), Q is finite. Hence we can hard-wire Q into a table and conclude that $A \in P$.

2. (q, r) computes a $\leq^{\mathrm{p}}_{\mathrm{btt}}$ reduction. We will prove the claim by induction on the number of queries. If the reduction does not ask any query, the claim is trivial. Assume that the claim is true for reductions asking fewer than k queries. We will prove the claim for reductions asking at most k queries. Take (q, r) that computes a $\leq^{\mathrm{p}}_{\mathrm{btt}}$ reduction and such that $|Q(x)| \leq k$, for all x. For any string x, let $m_x = \min\{|q| : q \in Q(x)\}$. We claim

that there exists an integer l such that for any x, if $m_x > l$ and $Q(x) = \{q_1, q_2, \ldots, q_{k'}\}$ then $r(\langle x, (q_1, 0), (q_2, 0), \ldots, (q_{k'}, 0)\rangle) = A(x)$. For contradiction assume that for any integer l, there exists x such that $m_x > l$ and $r(\langle x, (q_1, 0), (q_2, 0), \ldots, (q_{k'}, 0)\rangle) \neq A(x)$. Since A is recursive, for any l, we can find the lexicographically first x_l having such a property. All the queries in $Q(x_l)$ are longer than l and at least one of them should be in R_{K}. However, each of the queries can be described by $O(\log l)$ bits, which is the contradiction. Hence, there exists an integer l such that for any x, if $m_x > l$ then $r(\langle x, (q_1, 0), (q_2, 0), \ldots, (q_{k'}, 0)\rangle) = A(x)$. Thus we can encode the answers for all queries of length at most l into a table and reduce the number of queries in our reduction by one. Then we can apply the induction hypothesis.

3. (q, r) computes a $\leq^{\mathrm{p}}_{\mathrm{mtt}}$ reduction. q is computable in time n^c, for some $c > 1$. We claim that r does not depend on any query of length more than $2c \log n$. Assume that for infinitely many x, r does depend on queries of length more than $2c \log |x|$, i.e., if $Q(x) = \{q_1, q_2, \ldots, q_m\}$ and $a'_1, a'_2, \ldots, a'_m \in \{0, 1\}$ are such that $a'_i = 1$ if $q_i \in R_{\mathrm{K}}$ and $|q_i| \leq 2c \log |x|$, and $a'_i = 0$ otherwise, then $r(\langle x, (q_1, a'_1), (q_2, a'_2), \ldots, (q_m, a'_m)\rangle) \neq A(x)$. Since r is monotone, this may happen only for x that belong to A. The set of all such x can be enumerated, by assuming that all queries of length greater than $2c \log |x|$ are not in R_{K} and assuming that all shorter queries are in R_{K}, and then computing successively better approximations to the correct answers for the short queries by enumerating the complement of R_{K}, until an answer vector is obtained on which r evaluates to zero, although x is in A. Note that for better approximations to the true value of R_{K}, r will still evaluate to zero because r is a monotone reduction. Hence for given l, we can find the first x of length more than l in this enumeration. One of the queries in $Q(x)$ is of length more than $2c \log l$ and it belongs to R_{K}. But we can describe every query in $Q(x)$ by $c \log l + 2 \log \log l + \log l + O(1)$ bits, which is less than $2c \log l$. That is a contradiction. Since we have established that r depends only on queries of length at most $2c \log n$, we can encode information about all strings of this size that belong to R_{K} into a polynomially large table. Thus A is in P/poly. □

Theorem 7 *If A is recursive and it reduces to R_{K} via a polynomial-time $f(n)$-truth-table reduction then A is in* $\mathrm{P}/(f(n)2^{f(n)^3 \log f(n)})$.

Corollary 8 *If A is recursive and reduces to R_{K} via a polynomial-time truth-table reduction with $O(\log(n)/\log\log n)$ queries then A is in* P/poly.

Corollary 9 *Let $g(n)$ be such that $g(n)2^{g(n)^3 \log g(n)} < 2^n$. Then there exists a recursive A such that A does not reduce to R_{K} via a polynomial-time $g(n)$-truth-table reduction. In particular for any $\alpha < 1$ there exists a recursive A that does not reduce to R_{K} via a polynomial-time n^α-truth-table reduction.*

Proof of Theorem 7. W.l.o.g. $f(n)$ is unbounded. Let M be the reduction from A to R_{K} that uses at most $f(n)$ queries. Let $Q(x)$ be the query set that $M(x)$ generates.

We will remove from $Q(x)$ all the strings that have length at least $s_n = 2\log(f(n)) + 2\log\log f(n) + c$ for some suitably chosen constant c. Let $Q'(x) = Q(x)\bigcap\{0,1\}^{<s_n}$ be this reduced set.

Note that there are at most 2^{s_n} strings of length less than s_n and that there are at most $\binom{2^{s_n}}{f(n)} < (2^{s_n})^{f(n)} < 2^{f(n)3\log f(n)}$ possible subsets $Q'(x)$. Partition $\{0,1\}^n$ into equivalence classes, where $[x] = \{y : Q'(y) = Q'(x)\}$. We will show that for each equivalence class $[x]$ there is an answer sequence v_x such that, for all $y \in [x]$, y is in A if and only if M accepts y when the answers to $Q(y)$ are answered according to v_x for all of the queries in $Q'(y)$, and all of the long queries are answered negatively.

Thus the advice string consists of an encoding of v_x, which can be written using $f(n)$ bits, for each possible set $Q'(x)$. This yields the desired advice bound.

It remains only to show that the string v_x exists. Assume otherwise. Thus, given m, there is a recursive procedure that finds the lexicographically first string x of length n such that $\log f(n) > m$ and for all v there is some $y_v \in [x]$ on which the result of running $M(y_v)$ with answer vector v does *not* answer correctly about whether y_v is in A. Let v be the answer sequence for $Q'(x) \cap R_{\mathrm{K}}$, and let r be the number of 1's in v (i.e., r is the size of $Q'(x) \cap R_{\mathrm{K}}$). Thus, given (m, r) we can compute $Q'(x)$ and start the enumeration of the complement of R_{K} until we have enumerated all but r elements of $Q'(x)$. Thus we can compute v and find y_v. Since $M(y_v)$ is not giving the correct answer about whether y_v is in A, but M *does* give the correct answer when using R_{K} as an oracle, it follows that $Q(y_v)$ contains an element of R_{K} of length greater than s_n. However, this string is described by the tuple (m, r, i), along with $O(1)$ additional bits. For the appropriate choice of c, this has length less than s_n, which is a contradiction. □

Note that the preceding proof actually shows that, for every x such that $[x]$ has "small enough" Kolmogorov complexity, we can pick v_x to be the answer sequence for $Q'(x) \cap R_{\mathrm{K}}$. If this were true for *every* x, then it would follow easily that every decidable set A that is reducible to R_{K} via a polynomial-time truth-table reduction is in P/poly.

3.2 Pathological Universal Machines

Before presenting the results of this section, we digress in order to introduce some techniques that we will need.

The following development is motivated by a question that one can naturally ask: what is the size of $(R_{\mathrm{K}})^{=n}$? It is a part of folklore that the number of strings in R_{K} of length n is Kolmogorov random. But is it odd or even? One would be tempted to answer that since $|(R_{\mathrm{K}})^{=n}|$ is Kolmogorov random, the parity of it must also be random. The following universal Turing machine U_{even} shows that this is not the case.

Let U_{st} be the "standard" universal Turing machine. Consider the universal Turing machine U_{even} defined by: for any $d \in \{0,1\}^*$, $U_{\mathrm{even}}(0d) = U_{\mathrm{st}}(d)$ and $U_{\mathrm{even}}(1d) =$ the bit-wise complement of $U_{\mathrm{st}}(d)$. It is immediate that the size of $(R_{\mathrm{K}_{U_{\mathrm{even}}}})^{=n}$ is even for all n. To construct a universal Turing machine U_{odd} for which the size of $(R_{\mathrm{K}_{U_{\mathrm{odd}}}})^{=n}$ is odd for all n (large enough), is a little bit more complicated.

We will need the following definition. For any Turing machine U we can construct an *enumerator* (Turing machine) E that enumerates all pairs (d, x) such that $U(d) = x$, for $d, x \in \{0,1\}^*$. (The running time of E is possibly infinite.) Conversely, given an enumerator E that enumerates pairs (d, x) so that if (d, x) and (d, x') are enumerated then $x = x'$, we can construct a Turing machine U such that for any $x, d \in \{0,1\}^*$, $U(d) = x$ if and only if E ever enumerates the pair (d, x). In the following, we will often define a Turing machine in terms of its enumerator.

We define U_{odd} in terms of its enumerator E_{odd} that works as it is described below. E_{odd} will maintain sets of non-random strings $\{N_i\}_{i\in\mathbf{N}}$ during its operation. At any point in time, set N_i will contain non-random strings of length i that were enumerated by E_{odd} so far. E_{odd} will try to maintain the size of sets N_i to be odd (except while they are empty.)

Initialize all $\{N_i\}_{i\in\mathbf{N}}$ to the empty set.[2]
For all $d \in \{0,1\}^*$, run $U_{\mathrm{st}}(d)$ in parallel.
Whenever $U_{\mathrm{st}}(d)$ halts for some d and produces a string x do:
    Output $(0d, x)$.
    If $|0d| < |x|$ and $N_{|x|} = \emptyset$ then set $N_{|x|} := \{x\}$.
    Else if $|0d| < |x|$ and $x \notin N_{|x|}$ then:
        Pick the lexicographically first string y in $\{0,1\}^{|x|} - (N_{|x|} \cup \{x\})$.
        Set $N_{|x|} := N_{|x|} \cup \{x, y\}$ and output $(1d, y)$.
    Continue.
End.

It is easy to see that the Turing machine U_{odd} defined by the enumerator E_{odd} is universal. Also it is clear that for all n large enough, $(R_{\mathrm{K}_{U_{\mathrm{odd}}}})^{=n}$ is of odd size.

The ability to influence the parity of $(R_{\mathrm{K}_U})^{=n}$ allows us to (sparsely) encode any recursively enumerable information into R_{K_U}. We can state the following theorem.

Theorem 10 *For any recursively enumerable set A, there is a universal Turing machine U such that if $C = \{0^{2^{2^x}} : x \in A\}$, then $C \leq^{\mathrm{p}}_{\oplus\mathrm{tt}} R_{\mathrm{K}_U}$. Consequently, $A \leq^{EE}_{\oplus\mathrm{tt}} R_{\mathrm{K}_U}$.*

Proof. Observe, $C \subseteq \{0^{2^{2^i}} : i \in \mathbf{N}\}$. We will construct the universal Turing machine U so that for any integer $i > 3$, $0^{2^{2^i}} \in C$ if and only if $(R_{\mathrm{K}_U})^{=i}$ is of odd size. Then, the polynomial time parity reduction of C to R_{K_U} can be constructed trivially as well as the double-exponential parity reduction of A to R_{K_U}.

Let M be the Turing machine accepting the recursively enumerable set C. We will construct an enumerator E for U. It will work as follows. E will maintain sets $\{N_i\}_{i\in\mathbf{N}}$ during its computations. At any point in time, for every $i > 0$ the set N_i will contain

[2]We assume in the usual way that E_{odd} works in steps and at step s it initializes the s-th set of $\{N_i\}_{i\in\mathbf{N}}$ to the empty set. Our statements regarding actions that involve infinite computation should be interpreted in a similar way.

non-random strings of length i that were enumerated by E so far and E will try to maintain the parity of $|N_i|$ unchanged during most of the computation. E will also run M on all strings $z = 0^{2^i}$ in parallel and whenever some new string z will be accepted by M, E will change the parity of $N_{\log|z|}$ by making some new string of length $\log|z|$ non-random. The algorithm for E is the following.

Initialize all $\{N_i\}_{i\in\mathbf{N}}$ to the empty set.
For all $d \in \{0,1\}^*$ and $z \in \{0^{2^i} : i \in \mathbf{N}\}$, run $U_{\mathrm{st}}(d)$ and $M(z)$ in parallel.
Whenever $U_{\mathrm{st}}(d)$ or $M(z)$ halts for some d or $z = 0^{2^i}$ do:
  If $U_{\mathrm{st}}(d)$ halts and produces output x then:
    Output $(00d, x)$.
    If $|00d| < |x|$ and $x \notin N_{|x|}$ then:
      Pick the lex. first string y in $\{0,1\}^{|x|} - (N_{|x|} \cup \{x\})$.
      Set $N_{|x|} := N_{|x|} \cup \{x, y\}$ and output $(01d, y)$.
    Continue.
  If $M(0^{2^i})$ halts and $i > 3$ then:
    Pick the lexicographically first string y in $\{0,1\}^i - N_i$.
    Set $N_i := N_i \cup \{y\}$, and output $(1^{i-1}, y)$.
    Continue.
End.

Clearly, enumerator E defines a universal optimal Turing machine and for any integer $i > 3$, $0^{2^i} \in C$ if and only if $(R_{\mathrm{K}_U})^{=i}$ is of odd size. □

Parity is not the only way to encode information into R_{K}. The following theorem illustrates that we can encode the information so that one can use $\leq^{\mathrm{p}}_{\mathrm{dtt}}$ reductions to extract it. In particular, this proves our Theorem 1.

Theorem 11 *For any recursively enumerable set A, there is a universal Turing machine U such that if $C = \{0^{2^x} : x \in A\}$, then $\overline{C} \leq^{\mathrm{p}}_{\mathrm{dtt}} R_{\mathrm{K}_U}$. Consequently, $\overline{A} \leq^{EE}_{\mathrm{dtt}} R_{\mathrm{K}_U}$.*

Proof. First, define a universal Turing machine U_{opt} as follows: $U_{\mathrm{opt}}(0d) = U_{\mathrm{st}}(d)$ and $U_{\mathrm{opt}}(1d) = d$. Clearly, for any $x \in \{0,1\}^*$, $\mathrm{K}_{U_{\mathrm{opt}}}(x) \leq |x| + 1$. For any $d \in \{0,1\}^*$ and any $s \in \{0,1\}^5$, U is defined as follows:

On input $0ds$, run $U_{\mathrm{opt}}(d)$ and if $U_{\mathrm{opt}}(d)$ halts then output $U_{\mathrm{opt}}(d)s$.
On input $1d$ do:
  Run $U_{\mathrm{opt}}(d)$, until it halts.
  Let y be the output of $U_{\mathrm{opt}}(d)$.
  Check if $0^{2^{|y|}} \in C$.
  If $0^{2^{|y|}} \in C$ then output $y0^5$.
End.

It is clear that for any $x \in \{0,1\}^*$, $\mathrm{K}_U(x) \leq |x| + 2$. Further, for any $s, s' \in \{0,1\}^5 - \{0^5\}$, $K_U(xs) = K_U(xs')$. Finally, for any $y \in \{0,1\}^*$, $0^{2^{|y|}} \in C$ if and only if $K_U(y0^5) < K_U(y1^5) - 4$. Hence, if $0^{2^{|y|}} \in C$ then $y0^5 \notin R_{\mathrm{K}}$. The $\leq^{\mathrm{p}}_{\mathrm{dtt}}$ reduction of $\overline{C}$ to

R_{K} works as follows: on input 0^{2^n}, for all $y \in \{0,1\}^n$ ask queries $y0^5$. Output 0 if none of the queries lies in R_{K} and 1 otherwise. □

One could start to suspect that maybe all recursive functions are reducible to R_{K} in, say, doubly exponential time, regardless of which universal Turing machine is used to define R_{K}. We do not know if that is true but the following theorem shows that certainly disjunctive truth-table reductions are not sufficient.

Theorem 12 *For any computable time-bound $t(n) \geq n$, every set A in* $\mathrm{REC} \cap \bigcap_U \{A : A \leq_{\mathrm{dtt}}^{t(n)} R_{\mathrm{K}_U}\}$ *is computable in time $O(t^3(n))$.*

Theorem 5 is a corollary of Theorem 12.

Proof. It suffices to show that for each decidable set A that is not computable in time $O(t^3(n))$, there is a universal machine U such that A is not $\leq_{\mathrm{dtt}}^{t(n)}$-reducible to R_{K_U}. Fix a set A not computable in time $O(t^3(n))$.

Let U_{st} be a (standard) universal Turing machine, and define U so that for all d, $U(00d) = U_{st}(d)$. Note that, for every length m, fewer than $\frac{1}{4}$ of the strings of length m are made non-random in this way.

Now we present a stage construction, defining how U treats descriptions $d \notin \{00\}\{0,1\}^*$. We present an enumeration of pairs (d, y); this defines $U(d) = y$. In stage i, we guarantee that the i-th Turing machine q_i that runs in time $t(n)$ (in an enumeration of clocked Turing machines computing $\leq_{\mathrm{dtt}}^{t(n)}$ reductions) does not reduce A to R_{K_U}.

At the start of stage i, there is a length l_i with the property that at no later stage will any string y of length less than l_i be enumerated in our list of pairs (d, y). (At stage 1, let $l_1 = 1$.)

Let $\mathcal{T}$ be the set of all subsets of the strings of length less than l_i. For any string x, denote by $Q_i(x)$ the list of queries produced by the $\leq_{\mathrm{dtt}}^{t(n)}$ reduction computed by q_i on input x, and let $Q'(x)$ be the set of strings in $Q_i(x)$ having length less than l_i.

In Stage i, the construction starts searching through all strings of length l_i or greater, until strings x_0 and x_1 are found, having the following properties:

- $x_0 \notin A$,
- $x_1 \in A$,
- $Q'(x_1) = Q'(x_2)$, and
- One of the following holds
  - $Q_i(x_1)$ contains fewer than 2^{m-2} elements from $\{0,1\}^m$ for each length $m \geq l_i$, or
  - $Q_i(x_0)$ contains at least 2^{m-2} elements from $\{0,1\}^m$ for some length $m \geq l_i$

We argue below that strings x_0 and x_1 will be found after a finite number of steps.

If $Q_i(x_1)$ contains fewer than 2^{m-2} elements from $\{0,1\}^m$ for each length $m \geq l_i$, then for each string y of length $m \geq l_i$ in $Q_i(x_1)$, pick a different d of length $m-2$ and add the pair $(1d, y)$ to the enumeration. This guarantees that $Q_i(x_1)$ contains no element of R_{K_U} of length $\geq l_i$. Thus if q_i is to be a $\leq_{\mathrm{dtt}}^{t(n)}$ reduction of A to R_{K_U}, it must be the case that $Q'(x_1)$ contains an element of R_{K_U}. However, since $Q'(x_1) = Q'(x_0)$ and $x_0 \notin A$, we see that q_i is not a $\leq_{\mathrm{dtt}}^{t(n)}$ reduction of A to R_{K_U}.

If $Q_i(x_0)$ contains at least 2^{m-2} elements from $\{0,1\}^m$ for some length $m \geq l_i$, then note that at least one of these strings is not produced as output by $U(00d)$ for any string $00d$ of length $\leq m-1$. We will guarantee that U does not produce any of these strings on any description $d \notin \{00\}\{0,1\}^*$, and thus one of these strings must be in R_{K_U}, and hence q_i is not a $\leq_{\mathrm{dtt}}^{t(n)}$ reduction of A to R_{K_U}.

Let l_{i+1} be the maximum of the lengths of x_0, x_1 and the lengths of the strings in $Q_i(x_0)$ and $Q_i(x_1)$.

It remains only to show that strings x_0 and x_1 will be found after a finite number of steps. Assume otherwise. It follows that $\{0,1\}^*$ can be partitioned into a finite number of equivalence classes, where y and z are equivalent if both y and z have length less than l_i, or if they have length $\geq l_i$ and $Q'(y) = Q'(z)$. Furthermore, for the equivalence classes containing long strings, if the class contains both strings in A and in $\overline{A}$, then the strings in A are exactly the strings on which q_i queries at least 2^{m-2} elements of $\{0,1\}^m$ for some length $m \geq l_i$. This yields an $O(t^3(n))$-time algorithm for A, contrary to our assumption that A is not computable in time $O(t^3(n))$. □

Theorem 13 *For any computable time-bound $t(n) \geq n$, every set A in* $\mathrm{REC} \cap \bigcap_U \{A : A \leq_{\oplus\mathrm{tt}}^{t(n)} R_{\mathrm{K}_U}\}$ *is computable in time $O(t^3(n))$.*

Proof. It suffices to show that for each decidable set A that is not computable in time $O(t^3(n))$, there is a universal machine U such that A is not $\leq_{\oplus\mathrm{tt}}^{t(n)}$-reducible to R_{K_U}. In what follows we will describe such a machine U in terms of its enumerator E. Let $q_1, q_2, \ldots$ be an enumeration of all Turing machines (*query generators*) that work in time at most $t(n)$. During the construction of E we will diagonalize against all q_i's.

To diagonalize against machine q_i we will pick two strings $x_0 \notin A$ and $x_1 \in A$ and we will force the parity of $|q_i(x_0) \cap R_{\mathrm{K}_U}|$ and $|q_i(x_1) \cap R_{\mathrm{K}_U}|$ to be the same.

We will construct E so to maintain the parity of $|q_i(x_0) \cap R_{\mathrm{K}_U}|$ and $|q_i(x_1) \cap R_{\mathrm{K}_U}|$. To do so E will maintain sets $N_l, D_l, C_l, L_{l,j} \subseteq \{0,1\}^l$, for $l \in \mathbb{N}, j \in \{0,1\}$, where N_l will be the set of non-random strings that were seen so far and D_l will be the set of descriptions that were used so far to make some strings non-random. Sets N_l and D_l are initially empty. Sets C_l (the "common" queries of length l) and $L_{l,j}$ (the queries of length l asked on input x_i that are "left out" of the other query set) will be obtained by partitioning $q_i(x_0)$ and $q_i(x_1)$, for some i, and $|C_l - N_l|$ and $|L_{l,j} - N_l|$ will be maintained even.

Let c be such that for all $n \geq c$, $|\mathrm{K}_{U_{\mathrm{st}}}^{=n}| \geq 3$. E uses the following sub-procedure that can be invoked with any set of at least c strings, all having length l.

make-even(S):

Let l be the common length of strings in S.

If $|S - N_l|$ is even do nothing

Otherwise do the following:

Pick $x \in S - N_l$ and $d \in \{0,1\}^{l-2} - D_{l-2}$.

Set $N_l := N_l \cup \{x\}$ and $D_l := D_l \cup \{d\}$.

Output $(1d, x)$.

End.

E plays two strategies simultaneously.

The first strategy. For all $d \in \{0,1\}^*$, E runs $U_{\mathrm{st}}(d)$ in parallel. Whenever some computation $U_{\mathrm{st}}(d)$ halts and produces output x, E outputs $(00d, x)$. If $|0d| < |x|$ and $x \notin N_{|x|}$ then do the following: Set $l := |x|$. Set $N_l := N_l \cup \{x\}$ and if $C_l, L_{l,0}, L_{l,1}$ were already defined invoke *make-even*(C_l), *make-even*$(L_{l,0})$, *make-even*$(L_{l,1})$.

This strategy ensures that E determines a universal Turing machine and that $|C_l - N_l|$ and $|L_{l,j} - N_l|$ are maintained even. Note that procedure *make-even* will be forced to make some string non-random at most once per every string x that becomes non-random because of U_{st}. (In addition, it may be forced to make at most three additional strings of each length non-random when $L_{l,0}, L_{l,1}$ and C_l are defined.)

The second strategy. E proceeds according to the algorithm described below. The algorithm proceeds in stages. At stage k, it will diagonalize against reduction q_k.

Set $l_1 = c$.

For successive $k := 1, 2, 3, \ldots$, do the following:

Pick two strings $x_0 \notin A$ and $x_1 \in A$, each having length at least l_k, so that $q_k(x_0)^{\leq l_k} = q_k(x_0)^{\leq l_k}$.

As in the proof of Theorem 12, it is easy to argue that such strings exist.

Set $l_{k+1} := 1 + \max\{l_k, |y| : y \in q_k(x_0) \cup q_k(x_1)\}$.

For $i \in \{l_k, \ldots, l_{k+1} - 1\}$ and $j \in \{0,1\}$ do:

Set $L_{i,j} := (q_k(x_j) - q_k(x_{1-j})) \cap \{0,1\}^i$.

Set $C_i := q_k(x_0) \cap q_k(x_1) \cap \{0,1\}^i$.

Invoke *make-even*$(L_{i,j})$ and *make-even*(C_i).

Continue with the next k.

End.

It is clear from the construction that no q_i parity truth-table reduces A to R_{K_U}. $\square$

We conclude with the following observation that is a corollary to Kummer's result [Kum96].

Theorem 14 *For every universal Turing machine U and every time-constructible function $t(n) \geq n$, there is a recursive set $A \notin \mathrm{DSPACE}(t)$ such that $A \leq_{\mathrm{dtt}}^{\mathrm{p}} R_{\mathrm{K}_U}$.*

Proof. Fix any universal Turing machine U and time-bound $t(n) \geq n$. By Kummer's result, there is a time-bound t' such that the Halting problem dtt-reduces to R_{K_U} in time $t'(n)$. W.l.o.g. $t'(n) \geq n$. Let $A \notin \mathrm{DSPACE}(t(t'(2^n)))$ be a recursive set. Consider set $B = \{0^{t'(2^{|x|})-|x|-1}1x : x \in A\}$. Clearly, $B \notin \mathrm{DSPACE}(t(n))$. Since A is recursive, it reduces to R_{K_U} via a dtt-reduction running in time $t'(n^c)$, for some constant c. It follows that $B \leq^{\mathrm{p}}_{\mathrm{dtt}} R_{\mathrm{K}_U}$.

$\square$

4 Conclusions and Open Problems

Do there exist universal Turing machines U_1 and U_2 so that $R_{\mathrm{K}_{U_1}}$ and $R_{\mathrm{K}_{U_2}}$ are not recursively isomorphic? Or so that they are not in the same $\leq_m$-degree?

Can one show that not every decidable set is $\leq^{\mathrm{p}}_{\mathrm{tt}}$-reducible to R_{K} (at least for some choice of universal machine)?

Is there a proof of Hypothesis 4? It might be more feasible to prove a related hypothesis more in line with Theorems 6 and 7 of Section 3: For any universal machine: $\{A \in \mathrm{REC} : A \leq^{\mathrm{p}}_{\mathrm{T}} R_K\} \subseteq \mathrm{PSPACE/poly}$

5 Acknowledgments

We would like to thank to Kolya Vereshchagin and Troy Lee for helpful discussions.

References

[ABK+02] E. Allender, H. Buhrman, M. Koucký, D. van Melkebeek, and D. Ronneburger. Power from random strings. In *FOCS: IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 669–678, 2002.

[BH77] L. Berman and J. Hartmanis. On isomorphisms and density of NP and other complete sets. *SIAM Journal on Computing*, 6:305–323, 1977.

[Kum96] M. Kummer. On the complexity of random strings. In *Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 1046 of *Lecture Notes in Computer Science*, pages 25–36. Springer, 1996.

[Pos44] E. L. Post. Recursively enumerable sets of positive integers and their decision problems. *Bulletin of American Mathematical Society*, 50:284–316, 1944.

[Sch74] C. P. Schnorr. Optimal enumerations and optimal Gödel numberings. *Mathematical Systems Theory*, 8:182–191, 1974.