Power from 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

Dieter van Melkebeek[§]
University of Wisconsin, Madison, WI, USA
dieter@cs.wisc.edu

Detlef Ronneburger[¶]
York College, CUNY, Jamaica, NY, USA
ronnebg@york.cuny.edu

September 30, 2005

*This paper combines work presented at the *24th Conference on Foundations of Software Technology and Theoretical Computer Science* [All01] and at the *43rd IEEE Annual Symposium on Foundations of Computer Science* [ABK+02].

[†]Partially supported by NSF grant CCR-0104823.

[‡]Partially supported by NSF grant CCR-0104823. Part of this work was done while a graduate student at Rutgers University, NJ.

[§]Partially supported by NSF Career award CCR-0133693.

[¶]Partially supported by NSF grant CCR-0104823 and PSC-CUNY Award 60112-35 36. Part of this work was done while a graduate student at Rutgers University, NJ.

Abstract

We show that sets consisting of strings of high Kolmogorov complexity provide examples of sets that are complete for several complexity classes under probabilistic and non-uniform reductions. These sets are provably not complete under the usual many-one reductions.

Let $R_{\mathrm{C}}, R_{\mathrm{Kt}}, R_{\mathrm{KS}}, R_{\mathrm{KT}}$ be the sets of strings x having complexity at least $|x|/2$, according to the usual Kolmogorov complexity measure C, Levin's time-bounded Kolmogorov complexity Kt [Lev84], a space-bounded Kolmogorov measure KS, and a new time-bounded Kolmogorov complexity measure KT, respectively.

Our main results are:

1. R_{KS} and R_{Kt} are complete for PSPACE and EXP, respectively, under P/poly-truth-table reductions. Similar results hold for other classes with PSPACE-robust Turing complete sets.
2. $\mathrm{EXP} = \mathrm{NP}^{R_{\mathrm{Kt}}}$.
3. $\mathrm{PSPACE} = \mathrm{ZPP}^{R_{\mathrm{KS}}} \subseteq \mathrm{P}^{R_{\mathrm{C}}}$.
4. The Discrete Log, Factoring, and several lattice problems are solvable in $\mathrm{BPP}^{R_{\mathrm{KT}}}$.

Our hardness result for PSPACE gives rise to fairly natural problems that are complete for PSPACE under $\leq_{\mathrm{T}}^{\mathrm{p}}$ reductions, but not under $\leq_{\mathrm{m}}^{\log}$ reductions.

Our techniques also allow us to show that all computably enumerable sets are reducible to R_{C} via P/poly-truth-table reductions. This provides the first "efficient" reduction of the halting problem to R_{C}.

1 Introduction

Much recent work in derandomization can be viewed as an attempt to understand and exploit the interplay between the two common meanings of the phrase "random string": a string picked at random (according to some distribution), and a string with high Kolmogorov complexity (in some sense). In this paper, we further investigate the relationship between these two notions. We apply recent advances in derandomization to obtain fundamentally new types of complete sets for several standard complexity classes. The sets consist of random strings with respect to various Kolmogorov measures.

We will focus on the set $R_{\mathrm{C}} = \{x : \mathrm{C}(x) \geq |x|/2\}$ of strings with high traditional Kolmogorov complexity, as well as various resource-bounded variants R_μ for $\mu = \mathrm{KT}, \mathrm{KS}, \mathrm{Kt}$. See Section 2 for the definitions of KT, KS, and Kt. The choice of $|x|/2$ as a quantification of "high complexity" is rather arbitrary. Our results hold for any reasonable bound ranging from $|x|$ to $|x|^\epsilon$ for any positive ϵ. In most cases, our results carry over to other notions of resource-bounded Kolmogorov complexity that have been considered in the literature.

The sets R_μ of Kolmogorov random strings are good examples of sets with a lot of information content that is difficult to access. There are many examples in the literature where sets consisting of strings with high resource-bounded Kolmogorov complexity have been studied as possible examples of intractable sets that are not complete for any of the standard complexity classes. We list a few of these results here:

- Buhrman and Mayordomo [BM97] studied a time-bounded Kolmogorov complexity measure K^t for exponential time bounds $t(n) \geq 2^{n^2}$, and they showed that the set $R^t = \{x : \mathrm{K}^t(x) \geq |x|\}$ lies in $\mathrm{EXP} - \mathrm{P}$ and is *not* complete for EXP under polynomial-time Turing reducibility $\leq_{\mathrm{T}}^{\mathrm{p}}$.

- Ko [Ko91] studied the measure K^t for *polynomial* time bounds $t(n)$, and showed that the questions of whether the set $\{x : \mathrm{K}^t(x) \geq \epsilon|x|\}$ is in P or is coNP-complete (with respect to $\leq_{\mathrm{T}}^{\mathrm{p}}$ reductions) cannot be answered using relativizing techniques.

- Kabanets and Cai [KC00] study the Minimum Circuit Size Problem (MCSP) defined as the set of all pairs (χ_f, s) where χ_f denotes a string of length 2^n that is the truth-table of of a Boolean function f on n variables and s denotes an integer such that f can be computed by a Boolean circuit of size at most s. Because the circuit size of f is polynomially-related to $\mathrm{KT}(f)$ (see Section 2), MCSP can be seen to be related to R_{KT}. Kabanets and Cai present evidence that MCSP is not in P, but is also not likely to be NP-complete under $\leq_{\mathrm{m}}^{\mathrm{p}}$ reductions.

- In the traditional setting of Kolmogorov complexity, where no resource bounds are present, the set of Kolmogorov random strings R_{C} is easily seen to be co-c.e. and not decidable, but the complement of the halting problem is *not* reducible to R_{C} via a many-one reduction. It was shown only within the last decade by Kummer that a truth-table reduction is sufficient to reduce the complement of the halting problem to R_{C} [Kum96], although it had long been known [Mar66] that Turing reductions can be used. It should be emphasized that Kummer's reduction, which is in fact a disjunctive truth-table reduction, is *not* feasible and asks *many* queries. It can be shown that R_{C} is not complete for the co-c.e. sets under polynomial time disjunctive truth-table reductions. (This was originally observed in [ABK$^+$02]; a more general statement is proved in [ABK04].)

These results suggest that the sets of resource-bounded random strings are not complete for the complexity class they naturally live in. Buhrman and Torenvliet [BT01] gave some evidence that this is not the complete picture. They showed that for the *conditional* version of space bounded Kolmogorov complexity, the set of random strings is hard for PSPACE under NP reductions. However, their result had the major drawback that it needed conditional Kolmogorov complexity, used NP reductions, and, moreover, their proof technique could not be used beyond PSPACE.

Using very different techniques, we provide much stronger results in the same direction. We show that the set of random strings *can* be exploited by efficient reductions. For instance, we show that the set R_{Kt} of strings with high complexity using Levin's time-bounded Kolmogorov notion Kt [Lev84] is complete for EXP under truth-table reductions computable by polynomial-size circuits. Since this set is provably not complete under polynomial-time many-one reductions, we obtain natural examples that witness the difference in power of various reducibilities.

It is significant that all of our completeness results are obtained via derandomization techniques. This provides a new paradigm for proving completeness in settings where more traditional methods provably are not applicable.

In some instances, we are also able to provide completeness results under uniform reductions. By making use of multiple-prover interactive proofs for EXP [BFL91] we show that R_{Kt} is complete for EXP under NP-Turing reductions.

Of greater interest is the fact that the set R_{KS} of strings with high space-bounded Kolmogorov complexity is complete for PSPACE under ZPP-Turing reductions. Our proofs rely on the existence of complete sets for PSPACE that are both downward self-reducible and random self-reducible [TV02], and hence our proofs do not relativize. It remains unknown whether the results themselves hold relative to all oracles.

For the unbounded case we even show that PSPACE is reducible to R_{C} under *deterministic* polynomial time Turing reductions. The main tool to prove this is a polynomial time algorithm to construct a string of high Kolmogorov complexity, using R_{C} as an oracle.

For R_{KT}, a set in coNP, we do not obtain completeness results but we show that R_{KT} is hard under BPP-reductions for some well-known candidate NP-intermediate problems: Discrete Log, Factoring, and several lattice problems. These hardness results also hold for the minimum circuit size problem (MCSP), thereby improving results of [KC00].

1.1 The connection to derandomization

There is an obvious connection between Kolmogorov complexity and derandomization. For any randomized algorithm $\mathcal{A}$ having small error probability, and any input x, the coin flip sequences r resulting in a wrong answer are atypical, and hence have short descriptions of some kind.

By considering different notions of Kolmogorov complexity, less-obvious (and more-useful) connections can be exposed. Assume that the coin flip sequences that result in wrong answers have small μ-complexity, for some Kolmogorov measure μ. If we can generate a coin flip sequence r belonging to the set R_μ of strings with high μ-complexity, then we obtain an upper bound on the complexity of $\mathcal{A}$, by running the randomized algorithm on (x, r). Instead of generating a string in R_μ, if we assume only that we can check membership in R_μ, then we obtain a zero-error probabilistic algorithm, by picking a coin flip sequence r at random until we get one in R_μ (of which there are many), and then running $\mathcal{A}$ on (x, r).

The first interesting application of this approach is due to Sipser [Sip83]. His proof that BPP lies in the polynomial-time hierarchy uses $\mu = \mathrm{KD}^t$, the $t(n)$-time bounded distinguishing complexity, for some polynomial t. The corresponding set R_μ lies in Π_2^p. The hardness versus randomness tradeoffs by Babai et al. [BFNW93] and by Impagliazzo and Wigderson [IW97], and the "easy witness technique" of [Kab01, IKW02] can be cast as an application of this approach with $\mu = \mathrm{KT}$. A survey of some derandomization results from this perspective can be found in [All01]. The observation from [KvM02] that the construction of [IW97] relativizes with respect to any oracle A can be viewed in terms of a Kolmogorov measure which we denote as KT^A. This interpretation plays a crucial role in Trevisan's influential construction of extractors out of pseudorandom generators [Tre01].

Our main technique is to use relativizing hardness versus randomness tradeoffs in the contrapositive. Such results state that if there exists a computational problem in a certain complexity class $\mathcal{C}$ that is hard when given oracle access to A, then there exists a pseudorandom generator secure against A that is computable within $\mathcal{C}$. However, we argue that no pseudorandom generator computable in $\mathcal{C}$ can be secure against R_μ. Thus we conclude that every problem in $\mathcal{C}$ is easy given oracle access to R_μ, i.e., $\mathcal{C}$ reduces to R_μ. For our results, we exploit the nonuniform hardness versus randomness tradeoffs in [BFNW93] and [IW97], as well as the uniform ones in [HILL99] and [IW01].

1.2 The structure of complexity classes

The tools of reducibility and completeness are responsible for most of the success that complexity theory has had in proving (or providing evidence for) intractability of various problems. It has been known since the work of Ladner [Lad75] that, if P is not equal to NP, then there are intractable problems in NP that are not NP-complete. There are not many interesting candidates for this status, though. Certainly the sets constructed in [Lad75] are quite artificial – constructed by putting huge empty segments inside a standard complete problem such as SAT. Similarly, there are a great many notions of reducibility that have been considered and for many of these notions it is known that more powerful reducibilities provide more complete sets (at least for large complexity classes such as EXP) [Wat87, AAI+01]. However, almost all of the known constructions proceed by diagonalization and do not produce very "natural" languages.

We have already mentioned three notable examples that run counter to this trend [BM97, KC00, Ko91]. Using the insight that circuit size is closely connected to a version of time-bounded Kolmogorov complexity, all of these examples can be seen to be variations on a single theme. Our results amplify and extend these earlier papers. For instance, the set R^t (for $t = 2^{n^2}$) was shown in [BM97] not to be complete for EXP under $\leq_{\mathrm{T}}^{\mathrm{p}}$ reductions, but the authors presented no positive results showing how to reduce problems in EXP to R^t. In contrast, we show that this set is complete under $\leq_{\mathrm{tt}}^{\mathrm{P/poly}}$ and $\leq_{\mathrm{T}}^{\mathrm{NP}}$ reductions. To the best of our knowledge, this is the first example of a "natural" set A with the property that $\mathrm{P}^A \neq \mathrm{NP}^A$. All previous examples of such sets have been explicitly constructed via diagonalization, to separate P and NP. For polynomial time bounds t, Ko raised the prospect that R^t might be complete for NP under $\leq_{\mathrm{T}}^{\mathrm{SNP}}$ reductions, but he presented no unconditional results reducing supposedly-intractable problems to R^t. In contrast, we show that factoring and various problems arising in cryptography are reducible to R^t (and in many cases we present $\leq_{\mathrm{T}}^{\mathrm{ZPP}}$ reductions).

As an additional application of our techniques toward explicating the structure of complexity classes, we present natural examples of sets that witness the difference of $\leq_{\mathrm{tt}}^{\mathrm{p}}$ and $\leq_{\mathrm{T}}^{\log}$ or $\leq_{\mathrm{T}}^{\mathrm{p}}$ and $\leq_{\mathrm{m}}^{\mathrm{p}}$ completeness notions in PSPACE.

1.3 Outline of the rest of the paper

In Section 2 we present background and definitions regarding resource-bounded Kolmorogov complexity. In Section 3 we present our results for R_{Kt}, R_{KS}, and R_{C}, and in Section 4 those for R_{KT}. We conclude with open problems in Section 5.

2 Resource-Bounded Kolmogorov Complexity

In this section, we present the notions of resource-bounded Kolmogorov complexity that we use in stating most of our results. Before presenting our definitions, it is necessary to give some justification for introducing new definitions, because it might seem to the reader that the literature has more than enough notions of resource-bounded Kolmogorov complexity already.

2.1 Motivation

Kolmogorov complexity provides a very useful and elegant tool for measuring the complexity of finite objects. We refer the reader to [LV93] for the standard theorems and notation related to Kolmogorov complexity. Since the Kolmogorov complexity $C(x)$ of a string x cannot be computed, many definitions have been proposed for computable approximations to $C(x)$.

One of the most useful of these was given by Levin [Lev84]. Given a universal Turing machine U, Levin defined $\mathrm{Kt}(x|y)$ to be $\min\{|d| + \log t \ : \ U(d,y) = x \text{ in at most } t \text{ steps}\}$ and $\mathrm{Kt}(x)$ to be a shortcut for $\mathrm{Kt}(x|\lambda)$, where λ denotes the empty string. Levin's Kt measure has many useful properties, among which is the fact that it provides a search strategy for finding accepting computations for nondeterministic Turing machines that provably is nearly optimal. Stated another way: suppose that you want to find satisfying assignments for Boolean formulae. Given a formula

ϕ with n variables, you want to find a string $v \in \{0,1\}^n$ that satisfies ϕ. If this problem can be solved in time $t(n)$, then it can be solved in time nearly $O(t(n))$ by a program that, on input ϕ, enumerates strings v in order of increasing value of $\mathrm{Kt}(v|\phi)$. For details, see [LV93].

For reasons of ease of presentation, we will slightly change the formal definition of Kt in Section 2.2. The modification affects the value of Kt by at most a logarithmic additive term and does not alter any of Kt's desirable properties. The principal change is in the way the universal machine U uses the description d. In Levin's original definition, U uses the description d to output the entire string x. In the new definition, the description d allows U to compute each bit of x. That is, given d and i, U can compute the ith bit of x. Although this causes no substantial change to the measure Kt, it opens up the possibility of considering run times that are much smaller than the length of the string x. In turn, this allows us to introduce a new notion of time-bounded Kolmogorov complexity, denoted KT, which is very closely related to circuit complexity. More precisely, $\mathrm{KT}(x)$ is polynomially related to the minimum circuit size of the function that has x as its truth table (see Theorem 11).

Other notions of resource-bounded Kolmogorov complexity have been studied previously. One definition that one encounters fairly frequently ([Ko91, Lon86, LV93, BM97, BT01]) is the measure $\mathrm{K}^t(x)$, in which one explicitly bounds the running time of the universal machine U by $t(|x|)$, where the function t is an additional parameter. In contrast to our notion KT, there is no known relationship between circuit size and K^t. The need to fix the running time t also makes the notion K^t less robust with respect to the choice of the underlying universal machine U. See Section 2.2 for more details.

It turns out to be convenient to consider relativized measures KT^A where the universal machine is given access to an oracle A. This highlights some close connections with some previously-studied Kolmogorov complexity measures. For instance:

- Levin's measure $\mathrm{Kt}(x)$ is linearly-related to $\mathrm{KT}^A(x)$, where A is a certain complete set for E.
- The original Kolmogorov measure $\mathrm{C}(x)$ is linearly-related to $\mathrm{KT}^H(x)$, where H is the halting problem.

The relationship between KT and circuit size holds also in the relativized setting. Thus Levin's measure Kt is roughly the same thing as circuit size on oracle circuits that have access to an E-complete set, and C is roughly the same thing as circuit size relative to the halting problem.

The computational model that we use throughout this paper is the multi-tape Turing machine with random-access to its input tape. Our ideas work in any general model of computation. The main reason for providing random-access to the input tape is that $\mathrm{KT}(x|y)$ can be seen to be closely related to the circuit size of the function represented by x on circuits that have oracle access to the function represented by y. For this correspondence to hold, U must be able to access any bit of y quickly. Let us emphasize that all of our theorems in Sections 3 and 4 hold verbatim for models such as Turing machines with sequential access to their input tapes or Turing machines that have random access to all their tapes.

2.2 Formal Definitions

We begin with definitions relating to circuit size.

Definition 1 *We consider circuits with* AND , OR , *and negation gates, as well as oracle gates for one or more oracles. An oracle gate for an oracle A has inputs $z_1, \ldots, z_r$ for some r; it outputs the* *value 1 if the string $z = z_1 \dots z_r$ is in A, and outputs 0 otherwise. We define the size of a circuit as the number of connections in the circuit.*

We also associate circuit complexities with Boolean strings viewed as truth-tables of Boolean functions.

Definition 2 *Let x be a string in $\{0,1\}^*$ of length n. Let $2^{k-1} < n \leq 2^k$ and let x' be the string of length 2^k of the form $x0^i$ for some i. Let f_x be the Boolean function on k variables, having x' as its truth table. We define* $\mathrm{SIZE}(x)$ *to be the size of a minimal-sized circuit with k input variables computing the function f_x. For any oracle A,* $\mathrm{SIZE}^A(x)$ *denotes the circuit size required to compute f_x on circuits with oracle A. If $y \in \{0,1\}^*$, then* $\mathrm{SIZE}^{A,y}(x)$ *denotes the circuit size required to compute f_x on circuits with two kinds of oracle gates: oracle gates for A of arbitrary fan-in and oracle gates for f_y of fan-in $\lceil \log |y| \rceil$.*

For a good overview of circuit complexity refer to [Vol99].

We have already mentioned that we will be using Turing machines with random-access to its input tape; we now make more precise what we mean by this. We use essentially the same notion that was considered in [BIS90]. The machine has one read-only input tape of length n, a constant number of read-write working tapes of infinite length and a read-write input address tape. At every time step the machine can modify the content of its read-write tapes using the appropriate heads and move these heads left or right by one tape cell. It can also query the content of the input bit whose address is written on the input address tape. If there is no such input bit the reply to the query is the symbol '*'.

In the case where the machine is an oracle Turing machine, for each oracle the machine has one read-write oracle tape. At every step the machine can query any of its oracles whether the string written on the corresponding oracle tape belongs to the oracle set or not. We also allow finite oracles. For a finite oracle $y \in \{0,1\}^*$, the machine obtains as an answer to its query i bit y_i if $i \leq |y|$ and '*' otherwise. Note that the input tape behaves like an oracle tape accessing a finite oracle.

As usual, the *running time* of a Turing machine on a particular input is the number of steps before a computation of the machine stops on that input, and the *space* used during the computation is the number of read-write tape cells visited during the computation.

As with ordinary Turing machines, running time on our machines is closely related to circuit size.

Proposition 3 (Simulation of Circuits) *Let $k \geq 0$ be an integer. There is a constant c and a two-tape oracle Turing machine M such that for every oracle A, finite oracles $y_1, \dots, y_k$, and circuit $C^{A,y_1,\dots,y_k}$ of size m computing a function of n input bits, there is an encoding d_C of C of size $\leq cm(\log m + \log n)$, so that for any string x, if $|x| = n$ then $M^{A,d_C,y_1,\dots,y_k}(x)$ outputs $C^{A,y_1,\dots,y_k}(x)$ in time $\leq c(m^2 \log m + m \log n)$ and it outputs $*$ in time $\leq c(\log n)$ otherwise.*

Proposition 4 (Simulation by Circuits) *For every oracle Turing machine M there is a constant c_M such that if $M^{A,y_1,\dots,y_k}$ runs in time $t = t(n, n_1, \dots, n_k)$ on inputs of length n with oracles $y_1, \dots, y_k$ of length $n_1, \dots, n_k$, respectively, then for every n and $n_1, \dots, n_k$ there is a circuit $C_{n,n_1,\dots,n_k}$ of size $c_M t(t^2 + n + \log \sum_{i=1}^k n_i)$, such that $C^{A,y_1,\dots,y_k}(x) = M^{A,y_1,\dots,y_k}(x)$ for all inputs x of length n.*

Using the technique of of Hennie and Stearns ([HS66]) and Fürer [Für82, Für84] we can establish the following proposition.

Proposition 5 (Minimal Simulation Overhead) *There is a Turing machine U with 2 work tapes, such that for any oracle Turing machine M there is a constant c_M so that for any oracle A and finite oracles d and y and input x, there is a finite oracle d' of length at most $|d| + c_M$ such that $U^{A,d',y}(x) = M^{A,d,y}(x)$. The computation time of U is at most $c_M t \log t$ and the space used is at most $c_M s$, where $M^{A,d,y}(x)$ runs for time t and uses space s. Furthermore, if M is a two-tape machine, then the running time of U is bounded by $c_M t$.*

We call any machine U that satisfies the previous proposition a *universal* Turing machine; note that we require our universal Turing machines to be efficient in simulating other machines.

Definition 6 *A Turing machine U is* universal *if it satisfies all properties stated in Proposition 5.*

It is customary to define the Kolmogorov complexity function $\mathrm{C}(x)$ to be the length of the shortest description $d \in \{0,1\}^*$ such that $U(d) = x$, where U is a universal Turing machine. In order to present all of our definitions in a uniform framework, we give a slightly different definition below. We use the notation $x_{|x|+1} = *$ to denote the "end of string marker."

Definition 7 (Kolmogorov Complexity) *Let U be a Turing machine and let A be an oracle. Define*

$$\mathrm{C}_U^A(x|y) = \min\left\{ |d| : \forall b \in \{0,1,*\} \; \forall i \leq |x|+1 : U^{A,d,y}(i,b) \text{ accepts iff } x_i = b \right\}.$$

We let $\mathrm{C}_U(x|y)$ denote $\mathrm{C}_U^{\emptyset}(x|y)$, and we let $\mathrm{C}_U(x)$ denote $\mathrm{C}_U(x|\lambda)$.

Definition 7 deviates from the standard definition in two respects. First, the universal machine U gets the index i of a bit position of the string x as input and only needs to determine x_i, the ith bit of x. Since this mechanism alone does not encode the length of x, we stipulate that we obtain $*$ for bit position $i = |x| + 1$. As discussed in Section 2.1, this change allows us to consider sublinear running times for U, which will be critical for the new Kolmogorov measure KT we introduce.

Second, the way we define the value of x_i is through a distinguishing process, i.e., the machine U only needs to recognize the correct value of x_i. For the results in our paper, we could as well use the traditional defining mechanism, which calls for the machine U to produce x_i. In fact, we used that convention in our original definition of the measure KT [All01, ABK+02]. However, it has subsequently been useful to consider new measures in this framework, where nondeterministic and alternating machines U are also taken into account [AKRR03, All05]. In order that all of these measures can be presented in a uniform framework, we adopt the convention that the machine U only needs to recognize the correct value of x_i. That is, given description d, index i, and $b \in \{0,1,*\}$, U^d should accept (i,b) if and only if b equals x_i.

The reader can easily check that Definition 7 is exactly equivalent to the usual one. That is, if d is a description of x in the "traditional" definition, then d will also be a description of x in Definition 7, for a suitable machine U.

If U is a universal machine, we have the useful property that for any other machine U' there is a constant c such that $\mathrm{C}_U^A(x|y) \leq \mathrm{C}_{U'}^A(x|y) + c$ for all x, y and A. Following convention (as in

[LV93]), we pick one such universal machine U and define $\mathrm{C}^A(x|y)$ to be equal to $\mathrm{C}^A_U(x|y)$. $\mathrm{C}(x|y)$ and $\mathrm{C}(x)$ are defined similarly in terms of U.

Now we present a formal definition of Levin's time-bounded Kolmogorov complexity measure Kt using this framework.

Definition 8 (Kt) *Let U be a Turing machine and let A be an oracle. Define the measure $\mathrm{Kt}^A_U(x|y)$ of a string x as*

$$\mathrm{Kt}^A_U(x|y) = \min\Big\{ |d| + \log t : \ \forall b \in \{0,1,*\}\ \forall i \leq |x|+1 : \ U^{A,d,y}(i,b) \textit{ accepts in } t \textit{ steps iff } x_i = b \Big\}.$$

We use $\mathrm{Kt}_U(x|y)$ to denote $\mathrm{Kt}^{\emptyset}_U(x|y)$, and we use $\mathrm{Kt}_U(x)$ to denote $\mathrm{Kt}_U(x|\lambda)$.

Our definition is essentially equivalent to Levin's original one up to a logarithmic term. More precisely, for suitable choices of the underlying machines, Definition 8 yields a value that is between Levin's and Levin's plus $\log|x|$.[1]

As in the case of resource-unbounded Kolmogorov complexity, the measure Kt_U is essentially invariant under the particular choice of universal machine U. Now, however, instead of an additive constant term, it seems that we must accept an additive logarithmic term. Universal Turing machines can simulate any other Turing machine M in time $O(t\log t)$ (Proposition 5), where t is the original time required for the computation of M. We pick one such universal machine U and define $\mathrm{Kt}^A(x|y)$ to be equal to $\mathrm{Kt}^A_U(x|y)$. $\mathrm{Kt}(x|y)$ and $\mathrm{Kt}(x)$ are defined similarly in terms of U.

Another Kolmogorov measure that has been studied before is K^t, where the running time of the underlying machine U is bounded by a parameter t.

Definition 9 (K^t) *Let U be a Turing machine and t a time bound. Define the measure $\mathrm{K}^t_U(x)$ of a string x as*

$$\mathrm{K}^t_U(x) = \min\Big\{ |d| : \ \forall b \in \{0,1,*\}\ \forall i \leq |x|+1 : \ U^{d}(i,b) \textit{ accepts in } t(|x|) \textit{ steps iff } x_i = b \Big\}.$$

In contrast to all other Kolmogorov measures that we consider, changing the choice of universal Turing machine might cause a huge change in the value of $\mathrm{K}^t(x)$. Nonetheless, we follow the usual convention ([Ko91, Lon86, LV93, BM97, BT01]) and pick some universal machine U and define $\mathrm{K}^t(x)$ to be $\mathrm{K}^t_U(x)$. Proposition 5 guarantees that for any other choice U' of universal machine, there is a constant c such that $\mathrm{K}^{ct\log t}(x) \leq \mathrm{K}^t_{U'}(x)$ (although we observe that $\mathrm{K}^{ct\log t}(x)$ might be *much* less than $\mathrm{K}^t_{U'}(x)$). (In this paper, we will neither consider $\mathrm{K}^t(x|y)$ nor K^t relative to oracles A.)

It is now time to give the formal definition of the measure KT that will be one of our main objects of study. This definition can be seen as a variant on a definition introduced by Antunes, Fortnow, and van Melkebeek in [AFvM01].

[1] As discussed briefly in [LV93], Levin's original definition of Kt was formulated in terms of "Kolmogorov-Uspensky machines." This model of computation is one of several that allow for the existence of universal machines that can simulate any other algorithm with at most a linear slow-down; see also [BAJ00]. This allows for a more elegant statement of certain theorems regarding Kt complexity. For instance, using this model of computation, one can show that for a universal machine U, for every machine M $\mathrm{Kt}_U(x) \leq \mathrm{Kt}_M(x) + O(1)$, and searching for satisfying assignments y to a Boolean formula ϕ in order of increasing $\mathrm{Kt}(y|\phi)$ can be shown to be optimal up to a *linear* slow-down. However, Kolmogorov-Uspensky machines confer no special advantages when considering more refined measures such as KS and KT, and thus we choose to use the more familiar Turing machines in defining our measures.

Definition 10 (KT) *Let U be a Turing machine and A an oracle. Define the measure $\mathrm{KT}_U^A(x|y)$ to be*

$$\mathrm{KT}_U^A(x|y) = \min\left\{ |d| + t : \ \forall b \in \{0,1,*\} \ \forall i \leq |x|+1 : \ U^{A,d,y}(i,b) \text{ accepts in } t \text{ steps iff } x_i = b \right\}.$$

We omit the superscript A if $A = \emptyset$, and the string y if $y = \lambda$.

Note that the only difference between KT and Levin's measure Kt is that in KT the time bound is given exponentially more weight than in Kt. The capital "T" in the notation for KT reflects the fact that the time component weighs more strongly than in the Kt measure.

Along with the greater emphasis on running time, we unfortunately incur a greater sensitivity on the underlying machine U. Definition 6 implies that for any universal Turing machine U and any other machine U', $\mathrm{KT}_U^A(x|y) \leq c \cdot \mathrm{KT}_{U'}^A(x|y) \log \mathrm{KT}_{U'}^A(x|y)$. We pick one such universal Turing machine U (guaranteed to exist by Proposition 5) and define $\mathrm{KT}^A(x|y)$ to be equal to $\mathrm{KT}_U^A(x|y)$. $\mathrm{KT}(x|y)$ and $\mathrm{KT}(x)$ are defined similarly in terms of U.

If one views a given string x as the truth table of a function f_x, then $\mathrm{KT}(x|y)$ roughly corresponds to circuit complexity of f_x on circuits that have oracle access to f_y. This is made precise below.

Theorem 11 *There is a constant c such that for any oracle A and any strings x and y in $\{0,1\}^*$.*

1. $\mathrm{Size}^{A,y}(x) \leq c(\mathrm{KT}^A(x|y))^2 \left((\mathrm{KT}^A(x|y))^2 + \log|x| + \log|y|\right)$,
2. $\mathrm{KT}^A(x|y) \leq c\,(\mathrm{Size}^{A,y}(x))^2 \left(\log \mathrm{Size}^{A,y}(x) + \log\log|x|\right)$.

Proof.

1 Let c_1, c_2 be suitably large constants. Assume a given string x of length n has $\mathrm{KT}^A(x|y) = m$. Thus there is a description d_x of length at most m, such that $U^{A,d_x,y}(i,b)$ accepts iff $b = x_i$ in at most m steps. By Proposition 4, there is a circuit $C_{n,m}^{A,d_x,y}$ of size $c_1 m(m^2 + \log n + \log(m+|y|))$ such that $C_{n,m}^{A,d_x,y}(i,b) = U^{A,d_x,y}(i,b)$, for all i and b. The finite oracle d_x can be replaced by a circuit of size $c_2 m$ to obtain a desired circuit for x of size $cm^2(m^2 + \log n + \log(m+|y|))$, provided that $c \geq c_1 c_2$.

2 Let c_1 and M be the constant and machine given by Proposition 3. Assume that there is a circuit C of size m with oracle gates for A and y, such that on input i (in binary) the circuit computes $C(i) = x_i$ for $i \leq |x|$. Denote $n = |x|$.

By Proposition 3 there is an encoding d_C of C of size $c_1 m(\log m + \log\log n)$ such that machine M, given oracle access to A, d_C and y, outputs $C(i)$ in time $c_1 m^2 \log m + m \log\log n$. From M one can easily construct a two-tape machine M' such that $M'^{A,d_C,y}(i,b)$ accepts if $x_i = b$ and that runs in essentially the same time as M. Thus $\mathrm{KT}_{M'}^A(x|y) \leq c_2 m^2(\log m + \log\log n)$, for some $c_2 \geq c_1$.

The theorem now follows by Definition 6 and the fact that the defining Turing machine U for KT is universal.

□

So far we have considered time-bounded Kolmogorov complexity measures. In Section 3 we will see how these measures give rise to complete problems for time-bounded complexity classes. It is useful to define space-bounded notions that will yield complete problems for space-bounded complexity classes.

Definition 12 (KS) *Let U be a Turing machine and A be an oracle. We define the following space-bounded complexity measure.*

$$\mathrm{KS}_U^A(x|y) = \min\left\{ |d| + s : \forall b \in \{0,1,*\}\ \forall i \leq |x|+1 : U^{A,d,y}(i,b) \text{ accepts in } s \text{ space, iff } x_i = b \right\}.$$

We omit the superscript A if $A = \emptyset$, and the string y if $y = \lambda$.

As usual, we will fix a universal Turing machine U and then drop the subscript U when considering this space-bounded Kolmogorov complexity measure. Similar to the measure KT, the measure KS is somewhat sensitive to the exact choice of the universal machine U. However, regardless of the particular choice of universal Turing machine, Proposition 5 implies that for any Turing machine U', there is a constant c such that $\mathrm{KS}^A(x|y) \leq c\mathrm{KS}_{U'}^A(x|y)$.

As is the case with traditional Kolmogorov complexity, for each of these resource-bounded Kolmogorov complexity measures $\mathrm{KT}, \mathrm{KS}, \mathrm{Kt}$, every string has a description that is not much longer than itself:

Proposition 13 *There is a constant c, such that for all x: $\mathrm{KT}(x) \leq |x| + c\log|x|$, $\mathrm{KS}(x) \leq |x| + c\log|x|$, and $\mathrm{Kt}(x) \leq |x| + c\log\log|x|$.*

It is useful to observe that the measures C, Kt, and KS can be approximated in terms of KT^A, for appropriate choices of oracle A.

Theorem 14 *Let H denote the halting problem. There exist a complete set A for* E*, a complete set B for* $\mathrm{DSPACE}(n)$*, and a constant c such that*

- $\frac{1}{c}\mathrm{Kt}(x) \leq \mathrm{KT}^A(x) \leq c \cdot \mathrm{Kt}(x)$.
- $\frac{1}{c}\mathrm{KS}(x) \leq \mathrm{KT}^B(x) \leq c \cdot \mathrm{KS}(x)$.
- $\frac{1}{c}\mathrm{C}(x) \leq \mathrm{KT}^H(x) \leq c \cdot (\mathrm{C}(x) + \log|x|)$.

Proof. Let us prove first the relation between Kt and KT^A. The proof for KS is similar. Then we will consider the claim for C.

Let $A \in \mathrm{E}$ and let x be given, such that $\mathrm{KT}^A(x) = m$. Thus, there is a description d_x of length $|d_x| \leq m$, such that $U^{A,d_x}(i,b)$ accepts iff $x_i = b$ in time at most m. During the computation, U can ask queries of length at most m, and since $A \in \mathrm{E}$, each such query can be answered in time $2^{O(m)}$. If M denotes the algorithm that simulates the computation of $U^{A,d_x}(i,b)$ for every i by directly computing the answers to the oracle queries to A, then the description $d'_x = \langle M, d_x\rangle$ is sufficient for U to compute $U^{d'_x}(i,b)$ in time $2^{O(m)}$. As $|d'_x| = m + O(1)$, we can conclude that $\mathrm{Kt}(x) \leq m + O(1) + \log(2^{O(m)}) = O(m)$.

For the other inequality, let $A = \left\{ \langle (1^r, d_x), i, b\rangle : U^{d_x}(i,b) \text{ accepts in } 2^r \text{ steps} \right\}$. (It is not hard to show that A is complete for $\mathrm{DTime}(2^n)$ under linear-time many-one reductions.) Let x be given and let $\mathrm{Kt}(x) = m$. Thus, there is a description d_x of length m, such that $U^{d_x}(i,b)$ accepts iff $x_i = b$ in at most 2^m steps. The following algorithm M (computable by a two-tape Turing machine) accepts the input (i,b) iff $x_i = b$ and runs in time $O(m)$ given oracle A and given finite oracle $(1^m, d_x)$. Given (i,b), the machine M writes the query $q = \langle (1^m, d_x), i, b\rangle$ onto the query tape. If the oracle accepts q then M accepts, else it rejects. Clearly, given finite oracle $(1^m, d_x)$, the machine M accepts (i,b) iff $b = x_i$. The time required for M is $O(m + |d_x| + |q|) = O(m)$. The length of $|(1^m, d_x)|$ is $O(m)$. Thus, $\mathrm{KT}^A_M(x) = O(m)$. By Definition 6, the first part of the theorem follows.

Now we show that $\mathrm{C}(x) = O(\mathrm{KT}^H(x))$. Let $\mathrm{KT}^H(x) = m$. Thus there is a description d_x of length at most m, such that $U^{H,d_x}(i,b)$ accepts in time m iff $x_i = b$. Let r be the number of strings in H of length at most m. Note that r can be written using $O(m)$ bits. Let M be an algorithm with an oracle encoding d_x, r and m that, given (i,b) enumerates all the r elements of H that have length at most m and then simulates U^{H,d_x} on input (i,b). It follows that $\mathrm{C}_M(x) = O(m)$, which is sufficient to prove the claim.

It remains only to show that $\mathrm{KT}^H(x) = O(\,(\mathrm{C}(x) + \log|x|)\,\log(\mathrm{C}(x) + \log|x|))$. Let $\mathrm{C}(x) = m$ and let d_x be a description of x. The set $L = \{\langle d_x, i, b\rangle\ ;\ U^{d_x}(i,b) \text{ accepts }\}$ is a c.e. set; thus it is reducible to H in linear time by a trivial reduction f. Hence there is a machine M that, on input (i,b) with oracle d_x computes $f(d_x, i, b)$ and poses this question to the oracle H. The running time is $O(m + \log|x|)$. The claim follows by Definition 6. □

In [AKRR03], the approach developed here has been extended to provide new measures of resource-bounded Kolmogorov complexity that are polynomially related to branching program size and formula size.

2.3 Sets of Random Strings

Our attention will be focused on sets containing strings of high complexity, for various measures of complexity. Specifically, we consider the following sets:

Definition 15 *For any Kolmogorov complexity measure μ (such as* $\mathrm{C}, \mathrm{Kt}, \mathrm{KS}, \mathrm{KT}, \mathrm{K}^t$*), define* $R_\mu = \{x\ :\ \mu(x) \geq |x|/2\}$.

The bound of $|x|/2$ in the definition of R_μ is arbitrary; all of our results hold for any reasonable bound in the range $|x|$ to $|x|^\epsilon$ for any positive ϵ. Essentially, apart from the mere fact that strings in R_μ do not have short descriptions of the appropriate type, all we need is that the set R_μ has polynomial density. We refer to the *density* of a language L as the fraction of all strings of length n that belong to L. L is said to have *polynomial density* if it contains at least $2^n/n^k$ strings of each length n, for some k. The set of μ-incompressible strings, i.e., the set of strings x with $\mu(x) \geq |x|$, is well-known to have polynomial density for $\mu = \mathrm{C}$ [LV93]. The same holds for $\mu \in \{\mathrm{Kt}, \mathrm{KS}, \mathrm{KT}\}$ by a trivial counting argument: Since each of these measures μ involves a nontrivial additional cost on top of the length of the describing program d, μ-compressible strings of length n are defined by strings d of length less than $n-1$, of which there are fewer than 2^{n-1}. We refer the reader to [Kou03] for the detailed proofs of our main theorems in their full generality, for arbitrary ϵ.

Proposition 16 *The sets R_{C}, R_{Kt}, R_{K^t} for any t, and R_{KS} all have polynomial density.*

Many of our results about Kt and KT complexity carry over to K^t complexity for certain values of t because of the following observation. A similar statement holds for K^t in relationship to C and for KS in relationship to Kt.

Proposition 17 *For some constant $c > 1$,*

- $\mathrm{C}(x) \leq \mathrm{K}^t(x)$ *for any time bound t.*

- $\mathrm{K}^t(x) < \mathrm{KT}(x)$ *for any time bound* $t \geq n + c\log n$.
- $\mathrm{K}^t(x) < \mathrm{Kt}(x)$ *for any time bound* $t \geq (\log n)^c 2^n$.
- $\mathrm{Kt}(x) \leq c\mathrm{KS}(x)$.

Proof. The first item follows directly from the definition.

Let $m = \mathrm{KT}(x)$. Thus, there is a description d of length less than m such that each bit of x can be produced in time at most m. By Proposition 13, $m \leq n + c\log n$. Thus, the same description d shows that $\mathrm{K}^t(x) < m$ for any $t \geq n + c\log n$. The proof for Kt is analogous.

The last inequality follows from the fact that if a halting machine runs in time t and space s then $t \leq 2^{O(s)}$. □

Corollary 18 *For some constant* c,

- $R_{\mathrm{C}} \subseteq R_{\mathrm{K}^t}$ *for any time bound* t.
- $R_{\mathrm{K}^t} \subseteq R_{\mathrm{KT}}$ *for* $t \geq n + c\log n$.
- $R_{\mathrm{K}^t} \subseteq R_{\mathrm{Kt}}$ *for* $t \geq n^c 2^n$.

Proposition 19 *If* $\mathrm{Kt}(x) \leq |x|^\epsilon$, *then* $\mathrm{K}^{2^{n^\epsilon}}(x) \leq |x|^\epsilon$.

The following upper bounds on the complexity of the various sets of random strings are straightforward.

Proposition 20 $R_{\mathrm{C}} \in$ co-c.e., $R_{\mathrm{Kt}} \in \mathrm{E}$, $R_{\mathrm{KS}} \in \mathrm{DSPACE}(n)$, *and* $R_{\mathrm{KT}} \in \mathrm{coNP}$. *Also,* $R_{\mathrm{K}^t} \in \mathrm{EXP}$ *for any constructible* $t = 2^{n^{O(1)}}$, *and* $R_{\mathrm{K}^t} \in \mathrm{coNP}$ *for any constructible* $t = n^{O(1)}$.

We show in Corollary 32 that every c.e. set is reducible to R_{C} via a reduction computable by polynomial size circuits. It follows that R_{C} requires circuits of size at least 2^{n^ϵ} for some $\epsilon > 0$. For rest of these sets R_μ, the best unconditional circuit lower bound known is that none of these sets are in AC^0.

Theorem 21 *Let A be any set in AC^0 that contains at least $2^n/n^k$ strings of each length n. Then there is a constant c such that, for every length n, there is a string $x \in A \cap \{0,1\}^n$ such that* $\mathrm{KT}(x) \leq \log^c n$.

Proof. Nisan [Nis91] showed that the Nisan-Wigderson generator can be used to derandomize any probabilistic AC^0 circuit. His construction presents a generator G that takes a seed of length $\log^c n$ and produces pseudorandom output of length n, with the property that any AC^0 circuit C of size n^l and depth d that accepts at least $2^n/n^k$ inputs of length n must accept some string in the range of G. (The constant c depends on the constants l, k and d.) Furthermore, it was shown in [Vio04] that there is a circuit (actually, even a formula) of size $\log^{O(1)} n$ that takes s and i as input, and produces the ith bit of $G(s)$. It follows that, for any seed s, the pseudorandom string $G(s)$ has polylogarithmic KT complexity. □

Corollary 22 *For* $t = \Omega(n^2)$, *none of the sets* R_{Kt}, R_{K^t}, R_{KT}, *and* R_{KS} *are in* AC^0.

2.4 Nonhardness Results for $R\mu$

As stated in the introduction, earlier results had indicated that sets of strings with high resource-bounded Kolmogorov complexity would *not* be complete for the complexity classes in which they reside. In this subsection, we present a few of these non-hardness results as they relate to the sets R_{KT}, R_{KS}, and R_{Kt}.

Theorem 23
- *R_{Kt} is not hard for* EXP *under $\leq^{\mathrm{p}}_{\mathrm{tt}}$ reductions.*
- *R_{KS} is not hard for* PSPACE *under $\leq^{\log}_{\mathrm{T}}$ reductions.*

Proof. We present the proof for R_{KS}. The proof for R_{Kt} is analogous. (The interested reader can consult [All01].) Note first of all that $\leq^{\log}_{\mathrm{T}}$ reducibility coincides with $\leq^{\log}_{\mathrm{tt}}$ reducibility [LL76].

Let T be a subset of $\{0\}^*$ that is in PSPACE but not in L. Suppose $T \leq^{\log}_{\mathrm{tt}} R_{\mathrm{KS}}$, and let f be the query generator of such a reduction. Note that $\mathrm{KS}(f(0^n)) = O(\log n)$. Thus each of the strings y that the reduction queries on input 0^n has $\mathrm{KS}(y) = O(\log n)$. Hence, the only strings y for which the query can receive a "yes" answer are of length $O(\log n)$, and for such queries the answer is computable directly in space $O(\log n)$. Hence all of the queries can be answered in space $O(\log n)$ and it follows that $T \in \mathrm{L}$, contrary to our choice of T. □

An interesting picture emerges if we consider even more restrictive reducibilities than polynomial-time and logspace reductions. Most natural examples of NP-complete sets are complete even under many-one reductions computed by uniform AC^0 circuits. However, as the following theorem shows, uniform AC^0 reductions cannot reduce even some very small complexity classes to R_{KS} and R_{Kt}.

Theorem 24 *R_{Kt} and R_{KS} are not hard for TC^0 under AC^0 many-one reductions.*

Proof. Let A be any set that is hard for TC^0 under AC^0 many-one reductions. Agrawal shows in [Agr01] that A is also hard for TC^0 under length-increasing AC^0 reductions. Thus, there is a length-increasing AC^0 reduction f reducing 0^* to A. Since this reduction f is computable in logspace, it follows that $\mathrm{KS}(f(0^n))$ and $\mathrm{Kt}(f(0^n))$ are each $O(\log n)$. Since f is length-increasing, it follows that A must contain infinitely many strings of low KS and Kt complexity. □

Unfortunately, we do *not* know how to argue that the strings $f(0^n)$ in the preceding proof have low KT complexity; in fact it follows easily from Lemma 25 below that this happens if and only if every problem in the polynomial-time hierarchy has circuits of size $2^{n^{o(1)}}$.

The *Linear-time hierarchy* is a subclass of the polynomial-time hierarchy consisting of the union of the classes Σ_k-time(n).

Lemma 25 *Let s be a monotone nondecreasing function. Every problem in the linear-time hierarchy has circuits of size $s(n)^{O(1)}$ if and only if for each function f computable in AC^0, $\mathrm{KT}(f(0^n)) \leq s(\log n)^{O(1)}$.*

Proof. Suppose that every problem in the linear-time hierarchy has circuits of size $s(n)^{O(1)}$. Let f be computable in AC^0. By a standard padding argument (see, for instance [AG91]) it follows that the set $\{(n, i) :$ the ith bit of $f(0^n)$ is $1\}$ is in the linear-time hierarchy. It follows from our hypothesis that this set has circuits of size $s(\log n)^{O(1)}$, and therefore by Theorem 11 that $\mathrm{KT}(f(0^n)) \leq s(\log n)^{O(1)}$.

For the other direction, suppose that each function f computable in AC^0 has the property that $\mathrm{KT}(f(0^n)) \leq s(\log n)^{O(1)}$. Let A be any problem in the linear-time hierarchy, and consider the function f that maps 0^n for $n = 2^m$ to the characteristic string of A for inputs of length m. The function f is computable in AC^0. It follows from the hypothesis and Theorem 11 that A on inputs of size m has circuits of size $s(\log 2^m)^{O(1)} = s(m)^{O(1)}$. □

In particular, Lemma 25 states that SAT has polynomial-size circuits if and only if for every function f computable in uniform AC^0, $\mathrm{KT}(f(0^n))$ is polylogarithmic.

Based on Lemma 25, the best we can offer for R_{KT} is the following conditional non-hardness theorem. The hypothesis to this theorem is unlikely; thus the proper way to interpret this result is that it is unlikely that one will be able to prove that R_{KT} is hard for TC^0 under AC^0 reductions.

Theorem 26 *If every problem in the polynomial-time hierarchy has circuits of size $2^{n^{o(1)}}$, then R_{KT} is not hard for TC^0 under AC^0 many-one reductions.*[2]

Proof. Let A and f be as in the proof of Theorem 24. Since f is computable in AC^0, the hypothesis and Lemma 25 imply that $\mathrm{KT}(f(0^n)) \leq n^{o(1)}$. Since f is length-increasing, this means that A has to contain infinitely many strings of low KT complexity. □

On the other hand, if SAT requires large circuits, the proof of Lemma 25 ensures the existence of an AC^0 computable function f such that $f(0^n)$ has high KT complexity (for every large n that is a power of 2). By concatening elements of SAT with $f(0^n)$, we obtain a set SAT′ that is AC^0-isomorphic to SAT but where each element of SAT′ has large KT complexity. Combining the two directions, we see that SAT requiring large circuits is in fact equivalent to the existence of sets AC^0-isomorphic to SAT containing only strings of high KT complexity.

The results of this section are related to the issue that was raised by Kabanets and Cai [KC00] in considering the question of whether or not MCSP is NP-complete under length-increasing reductions. They observed that if MCSP (or R_{KT}) is complete for coNP under length-increasing reductions, then there is a polynomial-time computable length-increasing function f such that each string $f(0^n)$ has high KT complexity, which implies that there is an efficient way to construct the truth tables of certain functions on $m = \Theta(\log n)$ bits that require circuits of size $2^{\epsilon m}$ for some $\epsilon > 0$. As Kabanets and Cai discuss (see also [All01]), this implies BPP = P.

3 Complexity of R_{Kt}, R_{KS}, and R_{C}

Improving greatly the results in [BT01], we show that strings of high Kolmogorov complexity are very useful as oracles. Our main technique is to use relativizing hardness-randomness tradeoffs in the contrapositive. In particular we will argue that an appropriate set R_μ of Kolmogorov random strings can be used to distinguish the output of a pseudorandom generator G_f (based on a function f) from truly random strings. This in turn will enable us to efficiently reduce f to R_μ. In this section, we will exploit pseudorandom generators G_f that may take more time to compute than the randomized procedure they try to fool. They yield our hardness results for R_{Kt}, R_{KS}, and R_{C}. In the next section, we will use pseudorandom generators G_f that are more efficiently computable and obtain hardness results for R_{KT}.

[2]In [All01], it is asserted that the conclusion of this theorem follows from the weaker assumption that SAT has circuits of size $2^{n^{o(1)}}$. We do not know how to prove this stronger statement.

We first describe the derandomization tools we will use. Then we present some hardness results in terms of nonuniform (P/poly) reductions that apply to many complexity classes. Finally, we consider some special cases where we are able to strengthen our results to provide uniform reductions.

3.1 Tools

It will be useful to recall the definition of PSPACE-*robustness* [BFNW93]. A language A is PSPACE-robust if $\mathrm{PSPACE}^A = \mathrm{P}^A$. (Here we assume that machines are allowed to ask oracle queries of only polynomial size.) The complete sets for many large complexity classes like PSPACE, EXP, EXPSPACE, etc. have this property, as well as the complete sets (under linear-time reductions) for classes like DSPACE(n) and E. It will be useful for us to observe that the halting problem is also PSPACE-robust.

Theorem 27 *The halting problem H is* PSPACE-*robust.*

Proof. Let M be a PSPACE machine using space n^k.

The set $A = \{(1^n, i) :$ there are at least i strings in H of length at most $n\}$ is c.e., and thus it is reducible in linear time to H. Using binary search on calls to A, we can compute the number of strings in H of length at most n^k.

Now consider the set $B = \{(M, x, r) : M^{H'}(x)$ accepts, where H' is the set consisting of the first r strings of length at most $|x|^k$ that show up in the enumeration of $H\}$. Deciding whether M^H accepts x boils down to computing the desired number r (using calls to A) and then making a single call to B. Since both A and B are efficiently reducible to H, the proof is complete. □

We will use several related constructions that build a pseudorandom generator G_f out of a function f. They are all based on the Nisan-Wigderson paradigm [NW94]. The authors of [BFNW93] construct, for any $\epsilon > 0$, a variant $G_f^{\mathrm{BFNW}} : \{0,1\}^{n^\epsilon} \mapsto \{0,1\}^n$ such that for any x of size n^ϵ, the function $G_f^{\mathrm{BFNW}}(x)$ is computable in space $O(n^\epsilon)$ given access to the Boolean function f on inputs of size at most n^ϵ. Moreover, if f is PSPACE-robust, then there is a constant c independent of ϵ, such that each bit of $G_f^{\mathrm{BFNW}}(x)$ is computable in time $n^{\epsilon c}$ with oracle access to f. The following hardness versus randomness tradeoff holds.

Theorem 28 ([BFNW93, KvM02]) *Let f be a Boolean function, $\epsilon > 0$, and $G_f^{\mathrm{BFNW}} : \{0,1\}^{n^\epsilon} \mapsto \{0,1\}^n$ be the pseudorandom generator described above. Let T be a set and $p(n)$ a polynomial. If $|\Pr_{r \in U_n}[r \in T] - \Pr_{x \in U_{n^\epsilon}}[G_f^{\mathrm{BFNW}}(x) \in T]| \geq 1/p(n)$ for all large n, then there exists a polynomial size oracle circuit family $\{C_n\}_{n \in \mathbb{N}}$ with oracle T that computes f and queries T non-adaptively.*

In fact, as observed in [AKRR03], close analysis of [BFNW93, KvM02] reveals that the circuit family $\{C_n\}$ can be constructed to be in TC^0. Thus, in particular, we can also conclude that f is in L^T/poly.

In [IW01], Impagliazzo and Wigderson reexamine the approach of [BFNW93]. For any $\epsilon > 0$, their pseudorandom generator $G_f^{\mathrm{IW98}} : \{0,1\}^{n^\epsilon} \mapsto \{0,1\}^n$ is also computable in space $O(n^\epsilon)$ with oracle access to f on inputs of size at most n^ϵ. It satisfies the following uniform version of Theorem 28. Recall that a function f is *random self-reducible* if there exists a randomized polynomial-time reduction from f to itself such that each individual query the reduction makes on an input of length n is uniformly distributed among the inputs of length n. A function f is *downward self-reducible* if there exists a polynomial-time reduction from f to itself that only makes queries of length strictly less than the input.

Theorem 29 ([IW01]) *Let $f : \{0,1\}^* \mapsto \{0,1\}^*$ be a random and downward self-reducible function, $\epsilon > 0$, and $G_f^{\mathrm{IW98}} : \{0,1\}^{n^\epsilon} \mapsto \{0,1\}^n$ be the pseudorandom generator described above. Let T be a set and $p(n)$ be a polynomial. If $|\Pr_{r\in U_n}[r \in T] - \Pr_{x\in U_{n^\epsilon}}[G_f^{\mathrm{IW98}}(x) \in T]| \geq 1/p(n)$, for all n large enough, then there exists a probabilistic, polynomial time Turing machine with oracle T that on input x outputs $f(x)$ with probability at least $2/3$.*

The preceding two theorems provide the key derandomization techniques that are required to prove our completeness results. They are stated in the contrapositive of their original formulations since that is the way we will use them. However, some of our completeness results (namely for EXP and PSPACE) involve uniform reductions that make use of randomness. These reductions can then be further derandomized by applying hardness versus randomness tradeoffs in the standard way. We will make use of the strengthening of the results of [BFNW93], as provided by Impagliazzo and Wigderson [IW97] (see also [STV01]). Given access to a Boolean function f, they show how to construct a pseudorandom generator $G_f^{\mathrm{IW97}} : \{0,1\}^{O(\log n)} \mapsto \{0,1\}^n$ with the following property (see also [KvM02]):

Theorem 30 ([IW97, KvM02]) *For any $\epsilon > 0$, there exist constants $c, c' > 0$ such that the following holds. Let A be a set and $n > 1$ be an integer. Let $f : \{0,1\}^{c\log n} \mapsto \{0,1\}$ be a Boolean function that cannot be computed by oracle circuits of size $n^{c\epsilon}$ with oracle A. Then $G_f^{\mathrm{IW97}} : \{0,1\}^{c'\log n} \mapsto \{0,1\}^n$ satisfies: $|\Pr_{r\in U_n}[C^A(r) = 1] - \Pr_{x\in U_{c'\log n}}[C^A(G_f^{\mathrm{IW97}}(x)) = 1]| < 1/n$, for any oracle circuit C^A of size at most n.*

For x of size $c'\log n$, $G_f^{\mathrm{IW97}}(x)$ is computable in time polynomial in n given access to f on inputs of length $c\log n$.

3.2 Nonuniform Hardness Results

Our first result illustrates our main technique. Recall that a language L has polynomial density if it contains at least $2^n/n^k$ strings of each length n, for some k. We use the notation $A\leq_{\mathrm{tt}}^{\mathrm{P/poly}}B$ to denote that there exists a truth-table (i.e., nonadaptive) reduction from A to B that is computable by a family of polynomial-size circuits.

Theorem 31 *Let A be any* PSPACE*-robust set. Let L be a set of polynomial density such that for every $x \in L$, $\mathrm{KT}^A(x) > |x|^\gamma$ for some constant $\gamma > 0$. Then A is reducible to L via $\leq_{\mathrm{tt}}^{\mathrm{P/poly}}$ reductions.*

Proof. Let f be the characteristic function of A. Consider $G_f^{\mathrm{BFNW}} : \{0,1\}^{n^\epsilon} \mapsto \{0,1\}^n$, where we choose ϵ as follows. We know that every bit of G_f^{BFNW} is computable in time $n^{\epsilon c}$ for some constant c independent of ϵ, given access to A. We may assume that $c > 1$, so set $\epsilon = \gamma/2c$. We claim that any string in the range of G_f^{BFNW} has small KT^A complexity. Let $y = G_f^{\mathrm{BFNW}}(x)$, for some $x \in \{0,1\}^{n^\epsilon}$. On input x, every bit of $G_f^{\mathrm{BFNW}}(x)$ is computable in time $n^{\gamma/2}$ with access to oracle A. Hence, $\mathrm{KT}^A(y) \leq |x| + O(n^{\gamma/2}\log n) + O(1) \leq n^\gamma$. It follows that L distinguishes the output of G_f^{BFNW} from random, so by Theorem 28, f is $\leq_{\mathrm{tt}}^{\mathrm{P/poly}}$ reducible to L. □

By Theorem 14 and Proposition 16, we can apply Theorem 31 to the following choices for the pair (A, L):

- $(A, L) =$ (the set A from Theorem 14, R_{Kt}),
- $(A, L) =$ (the set B from Theorem 14, R_{KS}), and
- $(A, L) = (H, R_{\mathrm{C}})$.

(In order to apply Theorem 31 we require that A is PSPACE-robust. By Theorem 27, H is PSPACE-robust, and the sets A and B from Theorem 14 are clearly PSPACE-robust, since they are complete for EXP and PSPACE, respectively.) Together with Proposition 20, this leads to the following results.

Corollary 32
- *R_{Kt} is complete for* EXP *under $\leq_{\mathrm{tt}}^{\mathrm{P/poly}}$ reductions.*
- *R_{K^t} is complete for* EXP *under $\leq_{\mathrm{tt}}^{\mathrm{P/poly}}$ reductions for any constructible time bound t such that $2^{n^{\epsilon}} \leq t \leq 2^{n^{O(1)}}$ for some $\epsilon > 0$.*
- *R_{KS} is complete for* PSPACE *under $\leq_{\mathrm{tt}}^{\mathrm{P/poly}}$ reductions.*
- *$H \leq_{\mathrm{tt}}^{\mathrm{P/poly}} R_{\mathrm{C}}$.*

Proof. The results for R_{Kt}, R_{KS}, and R_{C} are immediate. The result for R_{K^t} follows from Proposition 19 and Theorem 31 (letting A be complete for E under linear-time reductions).

□

The last item in Corollary 32 appears to be the first "efficient" reduction from the halting problem to the Kolmogorov-random strings. This should be contrasted with the uniform but very high-complexity truth-table reduction of [Kum96].

A complete set for exponential space can be defined by considering a variant on KS complexity (which could naturally be denoted Ks), where the value to be minimized is $|d| + \log s$. Similarly, complete sets for doubly-exponential time can be defined by by minimizing $|d| + \log\log t$, etc.

It is natural to wonder how much non-uniform advice is needed, for the reductions of Corollary 32. For instance, can the reductions be done with linear advice? For the case of deterministic non-adaptive reductions to R_{Kt}, Ronneburger shows in [Ron04] that there is no fixed k such that advice of length n^k is sufficient, even if the reduction is not restricted to run in polynomial time, but instead can run for time 2^{n^k}. In contrast, in the next subsection we consider settings in which no nonuniform advice is needed at all.

3.3 Uniform Hardness Results

For the special cases of EXP and PSPACE, we are able to show hardness results under uniform notions of reducibility. First we state and prove a general uniform hardness result for PSPACE.

Theorem 33 *Let L be a set of polynomial density such that for every $x \in L$, $\mathrm{KS}(x) > |x|^{\gamma}$ for some constant $\gamma > 0$. Then* $\mathrm{PSPACE} \subseteq \mathrm{BPP}^L$.

Proof. We make use of the uniform derandomization technique of [IW01], together with the following theorem of [TV02].

Theorem 34 ([TV02]) *There exists a problem in* DSPACE(n) *that is hard for* PSPACE*, random self-reducible, and downward self-reducible.*

Let $f \in$ DSPACE(n) be a function that is downward and random self-reducible and that is hard for PSPACE, as guaranteed by Theorem 34. Consider $G_f^{\mathrm{IW98}} : \{0,1\}^{n^{\gamma/2}} \mapsto \{0,1\}^n$. Since the function G_f^{IW98} is computable in space $O(n^{\gamma/2})$ with oracle access to f on inputs of length $O(n^{\gamma/2})$ and f is computable in linear space, we can compute $G_f^{\mathrm{IW98}}(z)$ for $z \in \{0,1\}^{n^{\gamma/2}}$ in space $O(n^{\gamma/2})$. It follows that every y in the range of G_f^{IW98} satisfies $\mathrm{KS}(y) \leq O(|y|^{\gamma/2})$. Since L is a set of at least polynomial density that only contains strings y of KS complexity at least $|y|^\gamma$, L distinguishes the output of G_f^{IW98} from random strings and by Theorem 29, there is a probabilistic procedure with oracle L that on input x outputs $f(x)$ with probability at least 2/3. Hence, PSPACE $\subseteq$ BPP^L. □

For many sets L satisfying the hypothesis of Theorem 33 (including any such set that lies in PSPACE/poly, and all of the sets R_μ for which nonuniform hardness results were presented in the previous section), the BPP reduction of Theorem 33 can be improved to obtain a ZPP reduction. This is a consequence of the next lemma, which shows how probabilistic complexity classes can be derandomized using oracle access to strings of high Kolmogorov complexity. The proof makes use of the by-now familiar fact that having access to the truth table of a hard function can be used to derandomize BPP [NW94, IW97].

Lemma 35 *Let A be any oracle and L be a set such that $L \in \mathrm{P}^A/\mathrm{poly}$ and for every $x \in L$, $\mathrm{KT}^A(x) > |x|^\gamma$ for some constant $\gamma > 0$.*

1. $\mathrm{MA}^L = \mathrm{NP}^L$ *if L contains at least one string of almost every length.*
2. $\mathrm{BPP}^L = \mathrm{ZPP}^L$ *if L is of at least polynomial density.*
3. $\mathrm{BPP}^L = \mathrm{P}^L$ *if there exists a polynomial time machine with oracle access to L that, on input 0^n, produces an element of L of length n.*

All three statements hold even if the respective hypothesis on L holds only for infinitely many constants c and almost all lengths of the form n^c.

Proof. 1. The NP-machine M guesses a string $\chi_f \in L$ of length m, for $m = n^c$, and interprets it as the truth-table of a Boolean function f on inputs of size $\log m$. Since $\chi_f \in L$, $\mathrm{KT}^A(\chi_f) \geq m^\gamma$. Thus by Theorem 11, f requires circuits of size at least $\Omega(m^{\gamma/3})$ even when the circuit has access to the oracle A. If, instead of A, the circuit has access to L, then (because of our assumption that L is reducible to A by circuits of size m^k for some $k \geq 1$), the size required to compute f is at least $\Omega(m^{\gamma/6k})$.

The function f is of sufficient hardness to construct the generator G_f^{IW97}, as stated in Theorem 30, to stretch $O(\log n)$ random bits into n pseudorandom bits that are indistinguishable from random by any oracle circuit of size n with access to L. Thus we can fully derandomize the probabilistic part of the MA-computation.

2. The proof is almost identical to the preceding argument. We have the additional assumption that L has polynomial density. Thus, instead of "guessing" the string χ_f as in the preceding argument, we can repeatedly pick strings at random. The expected running time until we find a $\chi_f \in L$ is polynomial. The result follows.

3. Instead of randomly choosing a candidate for χ_f as in the proof of statement 2, we use the generating algorithm from the hypothesis to produce it. The rest of the argument is unchanged. $\square$

By Theorem 14 and Propositions 16 and 20, the common hypothesis of Lemma 35 holds for $L = R_{\mathrm{Kt}}$ (using the set A from Theorem 14 as the set A), for $L = R_{\mathrm{KS}}$ (using the set B from Theorem 14 as the set A), and for $L = R_{\mathrm{C}}$ (using $A = H$). Thus one can obtain, e.g., $\mathrm{BPP}^{R_{\mathrm{Kt}}} = \mathrm{ZPP}^{R_{\mathrm{Kt}}}$, $\mathrm{BPP}^{R_{\mathrm{KS}}} = \mathrm{ZPP}^{R_{\mathrm{KS}}}$, $\mathrm{NP}^{R_{\mathrm{Kt}}} = \mathrm{MA}^{R_{\mathrm{Kt}}}$. Moreover:

Corollary 36 $\mathrm{PSPACE} = \mathrm{ZPP}^{R_{\mathrm{KS}}} \subseteq \mathrm{ZPP}^{R_{\mathrm{Kt}}}$.

The techniques of Theorem 33 cannot be used to prove hardness results for classes larger than PSPACE. For EXP, however, we are able to use the theory of interactive proofs to obtain hardness results under NP-Turing reducibility. First, we mention a result that is implicit in [BFNW93].

Theorem 37 ([BFNW93]) *If* $\mathrm{EXP} \leq_{\mathrm{T}}^{\mathrm{P/poly}} L$*, then* $\mathrm{EXP} \subseteq \mathrm{MA}^L$.

Proof. Let A be any language in EXP. Then A is accepted by a 2-prover interactive proof system with provers computable in EXP [BFL91]. Thus these strategies are computable in $\mathrm{P}^L/\mathrm{poly}$. The MA^L protocol is as follows. Merlin sends Arthur the polynomial-size circuits that, when given access to the oracle L, compute the answers given by the two provers for A. Arthur then executes the MIP protocol, simulating the provers' answers by executing the circuits and querying the oracle L. $\square$

Corollary 38 *Let L be a set of polynomial density in* EXP *such that for every $x \in L$,* $\mathrm{Kt}(x) > |x|^\gamma$ *for some constant $\gamma > 0$. Then* $\mathrm{EXP} = \mathrm{NP}^L$.

Proof. The inclusion $\mathrm{NP}^L \subseteq \mathrm{EXP}$ is trivial. For the other inclusion, observe that Corollary 32 implies that the hypothesis to Theorem 37 is satisfied. The theorem now follows from the first item of Lemma 35. $\square$

Corollary 39 $\mathrm{EXP} = \mathrm{NP}^{R_{\mathrm{Kt}}}$.

We also obtain the following curious corollary.

Corollary 40 $\mathrm{P}^{R_{\mathrm{K}t}} \neq \mathrm{NP}^{R_{\mathrm{K}t}} = \mathrm{EXP}$ *for any constructible monotone t such that* $n^{O(1)} 2^n t(n-1) \leq t(n) \leq 2^{n^{O(1)}}$*, e.g.,* $t = 2^{n^2}$.

Proof. The equality follows from Corollary 38 by the third part of Proposition 17 and by Proposition 20. The inequality holds because the proof in [BM97] showing that R^t is not complete for EXP under $\leq_{\mathrm{T}}^{\mathrm{p}}$ reductions carries over unchanged to $R_{\mathrm{K}t}$. We include a somewhat simplified argument.

Consider a $\leq_{\mathrm{T}}^{\mathrm{p}}$ reduction from a unary language A to $R_{\mathrm{K}t}$. We claim that for inputs $x = 0^n$ of sufficiently large length, none of the queries of length n or more can be in $R_{\mathrm{K}t}$. Otherwise, we could describe the first such query q by n and the index i of the query. This yields a description of length $O(\log n) < n$. Moreover, we can generate q from this description in time $n^O(1) 2^n t(n-1) \leq t(n)$ by simulating the polynomial-time reduction up to the moment it generates its ith query and answering all earlier queries by running the trivial exponential-time algorithm for $R_{\mathrm{K}t}$. This contradicts the hypothesis that q is in R_{K^t}. Thus, we can decide A as follows: Run the reduction by answering queries of length n or more negatively, and running the trivial exponential-time algorithm for R_{K^t} on smaller queries. This yields a correct algorithm for A that runs in DTime(2^{dn}) for some fixed constant independent of A. However, the time hierarchy theorem implies that there are unary languages A in EXP that have higher complexity than this. $\square$

To the best of our knowledge, all prior examples of oracles for which P $\neq$ NP have been specifically constructed for that purpose. We know of no other "natural" examples of sets A for which $\mathrm{P}^A \neq \mathrm{NP}^A$.

We are unable to present any completeness or hardness results under deterministic polynomial-time reductions for any of the resource-bounded notions of Kolmogorov complexity that we have studied. It is interesting to observe, however, that we are able to do better when we consider the undecidable set R_{C}.

Theorem 41 PSPACE $\subseteq \mathrm{P}^{R_{\mathrm{C}}}$.

Theorem 41 follows immediately from Theorem 33 and the following lemma, which implies that $\mathrm{BPP}^{R_{\mathrm{C}}} = \mathrm{P}^{R_{\mathrm{C}}}$ by the third item of Lemma 35 applied to $A = H$ and $L = R_{\mathrm{C}}$. It is observed in [ABK04] that very similar techniques suffice to show that NEXP $\subseteq \mathrm{NP}^{R_{\mathrm{C}}}$.

Lemma 42 ([BFNV05]) *There is a polynomial-time algorithm with oracle access to R_{C} that, on input 0^n, outputs a string of length n in R_{C}.*

We include a simple proof of Lemma 42 which works for our standard version of R_{C}, i.e., with a randomness threshold of $|x|/2$. Our proof also works for the smaller randomness thresholds we consider but does not work for a randomness threshold close to $|x|$. [BFNV05] gives a more complicated proof for the case where the oracle used is $\{x : \mathrm{C}(x) \geq |x|\}$, i.e., for a randomness threshold of $|x|$; that proof works for any reasonable randomness threshold between $|x|$ and $|x|^\epsilon$ for any positive ϵ (see Remark 2 after Theorem 10 in [BFNV05]).

Proof of Lemma 42 for the standard randomness threshold of $|x|/2$. For any $m = n^{O(1)}$, we show how to construct a string z of length m such that $\mathrm{C}(z) \geq |z|/2$ via a polynomial-time computation with access to oracle R_{C}. We will use the following property referred to as symmetry of information in [LV93, Section 2.8]: There exists a constant c_1 such that for any strings z and y, $\mathrm{C}(zy) \geq \mathrm{C}(z) + \mathrm{C}(y|z) - c_1 \log |zy|$.

We build z inductively. We start with z equal to the empty string. Assume (inductively) that $\mathrm{C}(z) \geq |z|/2$. Try all strings y of length $2c_1 \log m$, and use the oracle R_{C} to see if $\mathrm{C}(zy) \geq |zy|/2$. We are guaranteed to find such a y, since $\mathrm{C}(y|z) \geq |y|$ holds for most y, and for any such y, $\mathrm{C}(zy) \geq \mathrm{C}(z) + \mathrm{C}(y|z) - c_1 \log |zy| \geq |z|/2 + |y| - c_1 \log |zy| \geq |zy|/2$. Set z to be zy. $\square$

In a similar way, part 3 of Lemma 35 would allow us to obtain a *deterministic* polynomial-time reduction of PSPACE to R_{KS} in Corollary 36 if there was a deterministic method to construct a string of high KS complexity using R_{KS} as an oracle. We do not know if this is possible.

The question of whether or not long elements of R_{Kt} can be obtained in polynomial time relative to R_{Kt} turns out to be of critical importance in understanding the completeness properties of R_{Kt}. Let us formalize this as follows. Let Q denote the proposition "For all c there is a function f computable in polynomial time relative to R_{Kt} such that $f(0^n) \in R_{\mathrm{Kt}} \cap \{0,1\}^{n^c}$." If Q is true, then Corollary 36 and the third item of Lemma 35 imply that PSPACE $\subseteq \mathrm{ZPP}^{R_{\mathrm{Kt}}} = \mathrm{P}^{R_{\mathrm{Kt}}}$; moreover

$R_{\mathrm{Kt}} \notin \mathrm{P}$ since otherwise Corollary 39 implies that $\mathrm{EXP} = \mathrm{NP}^{R_{\mathrm{Kt}}} = \mathrm{NP} \subseteq \mathrm{PSPACE} \subseteq \mathrm{P}^{R_{\mathrm{Kt}}} = \mathrm{P}$. On the other hand, if Q is false then $\mathrm{EXP} \neq \mathrm{P}^{R_{\mathrm{Kt}}}$. This follows from an argument similar to one presented in the proof of Corollary 40: If there is some c for which no computation in $\mathrm{P}^{R_{\mathrm{Kt}}}$ on input 0^n can access a string in R_{Kt} of length greater than n^c, then it follows that every unary language in $\mathrm{P}^{R_{\mathrm{Kt}}}$ is in $\mathrm{DTime}(2^{3n^c})$, which contradicts the time hierarchy theorem.

It should be noted in this regard that the property of symmetry of information, which was used in the proof of Lemma 42, provably does not hold for R_{Kt} [Ron04].

Let us mention one other consequence of our PSPACE-completeness result. Watanabe and Tang showed in [WT92] that $\leq_{\mathrm{T}}^{\mathrm{p}}$ and $\leq_{\mathrm{m}}^{\mathrm{p}}$ reducibilities yield different classes of PSPACE-complete sets if and only if $\leq_{\mathrm{T}}^{\mathrm{BPP}}$ and $\leq_{\mathrm{m}}^{\mathrm{p}}$ reducibilities yield different PSPACE-complete sets. Using similar techniques, we are able to prove the following.

Theorem 43 *There is a tally set T such that $\mathrm{PSPACE} = \mathrm{P}^{R_{\mathrm{KS}} \cup T}$, where at least one of the following holds:*

- *R_{KS} is complete for PSPACE under $\leq_{\mathrm{tt}}^{\mathrm{p}}$ reductions, but not under $\leq_{\mathrm{T}}^{\log}$ reductions, or*
- *$R_{\mathrm{KS}} \cup T$ is complete for PSPACE under $\leq_{\mathrm{T}}^{\mathrm{p}}$ reductions, but not under $\leq_{\mathrm{m}}^{\mathrm{p}}$ reductions.*

Proof. By Theorem 33 there is a probabilistic Turing machine M running in time n^k accepting QBF with oracle R_{KS}, with error probability less than 2^{-2n}. Let S be the set of all strings r such that r is the lexicographically-first string of length n^k with the property that, for all strings x of length n, $M^{R_{\mathrm{KS}}}(x, r)$ accepts iff $x \in \mathrm{QBF}$. The set S is in PSPACE and, for almost all n, S contains exactly one string of length n^k. We encode the "good" coin flip sequences of S in the tally set $T = \{0^{2n^k+i}$: the i-th bit of the unique string of length n^k in S is $1\}$. Note that R_{KS} can only contain a finite number of strings in 0^*. It is immediate that $T \in \mathrm{PSPACE}$ and that QBF is in $\mathrm{P}^{R_{\mathrm{KS}} \cup T}$.

If $R_{\mathrm{KS}} \cup T$ is not complete for PSPACE under $\leq_{\mathrm{m}}^{\mathrm{p}}$ reductions, then the second condition of the theorem holds.

On the other hand, if $\mathrm{QBF} \leq_{\mathrm{m}}^{\mathrm{p}} R_{\mathrm{KS}} \cup T$, then we can apply a standard argument of [Ber78]. Berman showed that if a length-decreasing self-reducible set A is $\leq_{\mathrm{m}}^{\mathrm{p}}$-reducible to a tally set, then $A \in \mathrm{P}$. A similar argument applied to the $\leq_{\mathrm{m}}^{\mathrm{p}}$ reduction from $A = \mathrm{QBF}$ to $R_{\mathrm{KS}} \cup T$ yields a $\leq_{\mathrm{tt}}^{\mathrm{p}}$ reduction from QBF to R_{KS}. We sketch the argument below for completeness. This, combined with Theorem 23, shows that the first condition of the theorem holds.

The modification of Berman's argument goes as follows. Let M denote the length-decreasing self-reduction of A, and f the reduction from A to $R_{\mathrm{KS}} \cup T$. We build a reduction tree starting from the given input x as the single node. In the kth phase, we apply M to the active leaves of length $|x| + 1 - k$. Those nodes that map under f to a string outside of 0^* are made inactive. For each integer $m \geq 0$, we choose a representative among all nodes (if any) which f maps to 0^m. We keep the representative active and make all the other nodes which f maps to 0^m inactive. This process finishes after at most $|x|$ phases. Since the number of active nodes is polynomially bounded, each step only involves a polynomial amount of work. The leaves which f maps to strings outside of 0^* constitute the truth-table queries we make to R_{KS}. Once we know the answers to these queries, we can determine the answers to all remaining nodes in the reduction tree from bottom to top. □

Note that in either case, we obtain a set that is $\leq_{\mathrm{T}}^{\mathrm{p}}$-complete but not $\leq_{\mathrm{m}}^{\log}$-complete for PSPACE, namely R_{KS} or $R_{\mathrm{KS}} \cup T$. It was already known (using the techniques of [Wat87]) that $\leq_{\mathrm{T}}^{\mathrm{p}}$ and $\leq_{\mathrm{m}}^{\log}$

reducibilities provide different classes of PSPACE-complete sets, but the preceding theorem provides fairly "natural" examples of sets witnessing the difference.

4 Complexity of R_{KT}

The previous section paints an illuminating picture about the hardness of sets with high Kt, KS, and C complexity for PSPACE, EXP, and larger complexity classes. In this section, we explore what these techniques have to say about the hardness of R_{KT}, a set in coNP. We are not able to show completeness of R_{KT} for coNP, but we can show the hardness of R_{KT} under randomized polynomial-time reductions for problems that are thought to be NP-intermediate: Discrete Log, Integer Factorization, and certain lattice problems.

We point out that the set R_{KT} seems closely related to the set MCSP defined in [KC00] as $\{(x, s) : \mathrm{SIZE}(x) \leq s\}$. Although we do not know of efficient reductions between R_{KT} and MCSP in either direction, all our hardness results for R_{KT} also hold for MCSP. Based on a connection with natural proofs, [KC00] showed that if MCSP is in P then, for any $\epsilon > 0$, there is a randomized algorithm running in time 2^{n^ϵ} that factors Blum integers well on the average. Our results imply that if MCSP is in P then there is a randomized polynomial-time algorithm that factors arbitrary integers.

4.1 Tools

In order to prove hardness results for R_{KT}, we need to apply hardness versus randomness tradeoffs for pseudorandom generators G_f of lower computational complexity than in Section 3. We will use the cryptographic pseudorandom generators that developed out of the seminal work by Blum and Micali [BM84], and by Yao [Yao82]. In [HILL99], it is shown how to construct such a pseudorandom generator $G_f^{\mathrm{HILL}} : \{0,1\}^n \mapsto \{0,1\}^{2n}$ out of any function f. The pseudorandom generator G_f^{HILL} is computable in polynomial time given access to f, and is secure provided f is one-way.

The known hardness versus randomness tradeoffs for G_f^{HILL} differ in two relevant respects from those used in Section 3. First, breaking G_f^{HILL} only lets us invert f on a nonnegligible fraction of the inputs, but not necessarily on all inputs. The implicit or explicit random self-reducibility of the problems considered in Section 3, allowed us there to make the transition from "nonnegligible fractions of the domain" to "the entire domain". However, unlike for EXP and PSPACE, there are no NP-complete problems that are known to be random self-reducible. In fact, there provably are no nonadaptively random self-reducible NP-complete sets unless the polynomial-time hierarchy collapses [FF93].

Nevertheless, for some specific NP-intermediate problems h and polynomial-time computable functions f (where h may or may not coincide with f^{-1}), a worst-case to average-case connection *is* known. That is, inverting f on a nonnegligible fraction of the inputs allows one to compute h efficiently on any input. We are able to prove hardness of R_{KT} for such problems.

The second difference with Section 3 is that the uniform hardness versus randomness tradeoffs in [HILL99] are as strong as the nonuniform ones. Therefore, unlike in Section 3, we only need to consider uniform reductions here.

We will apply G_f^{HILL} for functions f that take some additional parameters y besides the actual input x. In the case of the Discrete Log problem, for example, y will describe the prime p and the generator g of $\mathbb{Z}_p^*$ defining the basis for the logarithm. More precisely, we will consider functions of the form $f(y,x)$ that are length-preserving for every fixed y, i.e., $f_y : \{0,1\}^n \mapsto \{0,1\}^n$, where $f_y(x) = f(y,x)$. The function $f(y,x)$ will be computable uniformly in time polynomial in $|x|$ (so without loss of generality, $|y|$ is polynomially bounded in $|x|$). The hardness versus randomness tradeoff in [HILL99] can be stated as follows.

Theorem 44 ([HILL99]) *Let $f(y,x)$ be computable uniformly in time polynomial in $|x|$. For any oracle L, polynomial-time probabilistic oracle Turing machine M, and polynomial p, there exists a polynomial-time probabilistic oracle Turing machine N and polynomial q such that the following holds for any n and y: If*

$$|\Pr_{|r|=2n,s}[M^L(y,r,s)=1] - \Pr_{|x|=n,s}[M^L(y,G^{\text{HILL}}_{f_y}(x),s)=1]| \geq 1/p(n),$$

then

$$\Pr_{|x|=n,s}[f(y,N^L(y,f(y,x),s)) = f(y,x)] \geq 1/q(n),$$

where r and x are chosen uniformly at random and s denotes the internal coin flips of M or N, respectively.

Theorem 44 states that if there exists a distinguisher with access to an oracle L that distinguishes the output of G^{HILL}_f from the uniform distribution, then oracle access to L suffices to invert f on a polynomial fraction of the inputs. We now argue that such a distinguisher exists in the case where L denotes R_{KT} or any set of polynomial density that contains no strings of small KT complexity.

Theorem 45 *Let L be a language of polynomial density such that, for some $\epsilon > 0$, for every $x \in L$, $\text{KT}(x) \geq |x|^\epsilon$. Let $f(y,x)$ be computable uniformly in time polynomial in $|x|$. There exists a polynomial-time probabilistic oracle Turing machine N and polynomial q such that for any n and y:*

$$\Pr_{|x|=n,s}[f(y,N^L(y,f(y,x),s)) = f(y,x)] \geq 1/q(n),$$

where x is chosen uniformly at random and s denotes the internal coin flips of N.

Proof. Let $G_y(x)$ denote $G^{\text{HILL}}_{f_y}(x)$. As in [RR97], we use the construction of [GGM86] to modify G_y and obtain a pseudorandom generator G'_y producing longer output. Specifically, the proof of Theorem 4.1 of [RR97] constructs $G'_y : \{0,1\}^n \mapsto \{0,1\}^{2^k}$, and shows that if $z = G'_y(x)$, then $\text{SIZE}(z) = (n+k)^{O(1)}$. We can pick $k = O(\log n)$ such that for each x and polynomially bounded y, $\text{KT}(G'_y(x)) < |G'_y(x)|^\epsilon$. Thus L is a statistical test that accepts many random strings of length $|x|^{O(1)}$, but rejects all pseudorandom strings. As in [RR97], this gives us a probabilistic oracle machine M using L that distinguishes G_y from the uniform distribution. We then apply Theorem 44. □

4.2 Hardness Results

We now apply Theorem 45 to obtain our hardness results for R_{KT}. As will be clear from the proofs, R_{KT} can be replaced by any suitably dense language containing no strings of KT-complexity less than n^ϵ for some $\epsilon > 0$, such as R_{K^t} for polynomial t. Other examples of such sets can be found in P^{MCSP}, and thus our hardness results carry over to MCSP as well. As observed in [AKRR03], the reductions in this section show that, under popular cryptographic assumptions, it is impossible even to approximate $\mathrm{KT}(x)$ or $\mathrm{SIZE}(x)$ within $|x|^{1-\epsilon}$.

We first consider the Discrete Logarithm Problem, which takes as input a triple (p, g, z) where p is a prime number, $0 < g < p$, and $0 < z < p$, and outputs a positive integer i such that $g^i \equiv z \bmod p$, or 0 if there is no such i. We have:

Theorem 46 *The Discrete Logarithm Problem is computable in* $\mathrm{BPP}^{R_{\mathrm{KT}}}$.

Proof. On input (p, g, z), we first check if p is prime [AKS04].

Let n be the number of bits in p, and y denote the pair (p, g). Consider the function $f(y, x) = g^x \bmod p$. Theorem 45 with $L = R_{\mathrm{KT}}$ gives us a polynomial-time probabilistic oracle Turing machine N such that for some polynomial q and all p and g, with probability at least $1/q(n)$ over randomly chosen x and s, $N^{R_{\mathrm{KT}}}(p, g, g^x, s)$ produces an output i such that $g^i \equiv g^x \bmod p$.

Now we make use of the self-reducibility properties of the Discrete Log. In particular, on our input (p, g, z), we choose many more than $q(n)$ values v at random and run algorithm N on input $(p, g, zg^v \bmod p)$. If z is in the orbit of g, then with high probability at least one of these trials will return a value u such that $g^u = zg^v \bmod p$, which means that we can pick $i = u - v$ and obtain $z \equiv g^i \bmod p$.

On the other hand, if none of the trials is successful, then with high probability z is not in the orbit of g and the algorithm should return 0. □

We are not able to improve our reduction from a BPP-reduction to a ZPP-reduction. I.e., we know of no analog of Lemma 35 for KT-complexity. We note that, for inputs (p, g, x) such that x is in the orbit of g, which is the usual class of inputs for which the discrete log is of interest, we do have ZPP-like behavior, since we can check whether the number i we obtain satisfies $g^i \equiv x \bmod p$. However, when x is not in the orbit of g, we obtain no *proof* of this fact – merely strong evidence.

The proof of Theorem 46 relies on the random self-reducibility of the Discrete Log Problem. In the terminology of Section 4.1, this allows us to consider f^{-1} as the problem h that reduces to R_{KT}. The next problem we consider is a first example where h differs from f^{-1}.

Theorem 47 *Integer Factorization is computable in* $\mathrm{ZPP}^{R_{\mathrm{KT}}}$.

Proof. Consider Rabin's candidate one-way function $f(y, x) = x^2 \bmod y$, where $0 \leq x < y$. Theorem 45 with $L = R_{\mathrm{KT}}$ gives us a probabilistic polynomial-time procedure with oracle access to R_{KT} that, for any fixed y, finds an inverse of $f(y, x)$ for a fraction at least $1/|y|^{O(1)}$ of the x's with $0 \leq x < y$. Rabin [Rab79] showed how to use such a procedure and some randomness to efficiently find a nontrivial factor of y in case y is composite. This leads to a $\mathrm{BPP}^{R_{\mathrm{KT}}}$ algorithm for factoring. Since primality is in P [AKS04], we can avoid errors and obtain the promised $\mathrm{ZPP}^{R_{\mathrm{KT}}}$ factoring algorithm. □

Since Ajtai's seminal paper [Ajt04], several worst-case to average-case connections have been established for lattice problems. We will exploit these next.

We first review some lattice terminology. We refer to [Cai99] for more background. Given a set B of n linearly independent vectors $b_1, \ldots, b_n$ over $\mathbb{R}^n$, the set $L(B) = \{\sum_{i=1}^n z_i b_i \mid z_i \in \mathbb{Z}\}$ is called the lattice spanned by the basis B. We consider the usual notion of length of a vector v, $|v| = \sqrt{\sum v_i^2}$, and define the length of a set of vectors as the length of a longest vector in that set. For a lattice $L(B)$, $\lambda_i(B)$, $1 \leq i \leq n$, denotes the length of a shortest set of i linearly independent vectors from $L(B)$. The covering radius of $L(B)$, $\rho(B)$, is defined as the smallest ρ such that the union of all spheres of radius ρ centered at the points of $L(B)$ covers the entire space $\mathbb{R}^n$.

Applying our technique to Ajtai's worst-case to average-case connections and their subsequent improvements and extensions [CN97, Mic02], we obtain the following hardness results for R_{KT}.

Theorem 48 *For every $\epsilon > 0$, we can solve each of the following problems in* $\mathrm{BPP}^{R_{\mathrm{KT}}}$*: Given a basis $B \subseteq \mathbb{Z}^n$ (and a vector $t \in \mathbb{Z}^n$), find*

- *(Shortest Independent Vector Problem) a set of n linearly independent vectors in $L(B)$ of length at most $n^{3+\epsilon} \cdot \lambda_n(B)$,*
- *(Shortest Basis Problem) a basis for $L(B)$ of length at most $n^{3.5+\epsilon} \cdot \lambda_n(B)$,*
- *(Length of Shortest Vector Problem) a value $\tilde{\lambda}$, s.t. $\lambda_1(B) \le \tilde{\lambda} \le \omega(n^{3.5}\log n) \cdot \lambda_1(B)$,*
- *(Unique Shortest Vector Problem) a shortest vector in $L(B)$ in case $\lambda_2(B) > n^{4+\epsilon} \cdot \lambda_1(B)$,*
- *(Closest Vector Problem) a vector $u \in L(B)$ such that $|u - t| \le n^{3.5+\epsilon} \cdot \lambda_n(B)$, and*
- *(Covering Radius Problem) a value $\tilde{\rho}$, s.t. $\rho(B) \le \tilde{\rho} \le \omega(n^{2.5}\log n) \cdot \rho(B)$.*

The approximation factors in Theorem 48 represent the current state-of-the-art but are likely to be improved in the future.

Proof of Theorem 48. In order to construct the reduction for the SIVP (the Shortest Independent Vector Problem), we will use the following worst-case to average-case connection.

Theorem 49 ([Ajt04, CN97]) *For any $\epsilon > 0$, there exist functions $q(n) = n^{O(1)}$ and $m(n) = n\log^2 q$ with q a power of 2 such that the following holds. For every $c > 0$, if there is probabilistic algorithm A, such that, with probability at least $1/n^c$ (over $M \in \mathbb{Z}_q^{n\times m}$ and over the random choices of A), $A(M)$ outputs a nonzero vector $u \in \mathbb{Z}^m$ such that $|u| \le m$ and $Mu \equiv 0 \bmod q$, then there exists a* BPP^A*-algorithm A' that given any basis $B \subseteq \mathbb{Z}^n$, outputs with high probability a set of n linearly independent vectors in $L(B)$ of length at most $n^{3+\epsilon} \cdot \lambda_n(B)$.*

Let $q(n)$ and $m(n)$ be as in Theorem 49. Consider the collection of functions $f_{q,m} : \mathbb{Z}_q^{n\times m} \times \{0,1\}^m \mapsto \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^n$ defined by $f_{q,m}(M,v) = (M, Mv \bmod q)$. Letting $y = (q,m)$ and $x = (M,v)$, the function $f_y(x)$ is computable uniformly in time $|x|^{O(1)}$. By Theorem 45, there exists a polynomial-time probabilistic algorithm N with oracle access to R_{KT} that computes a preimage of $f_{q,m}(M,v)$ for at least a $1/n^{O(1)}$ fraction of the inputs (M,v).

For any fixed M, f maps to at most q^n different values. Thus there can be at most q^n vectors v, such that $f(M,v) \ne f(M,v')$ for all $v' \ne v$. In other words, for a fraction at least $1 - q^n/2^m > 1 - O(1/2^n)$ of the vectors v, there is a $v' \ne v$ with $f(M,v') = f(M,v)$. Therefore, if we pick M and v uniformly at random, and run N to invert $f(M, Mv \bmod q)$, with probability at least $1/n^{O(1)}$ we obtain a vector $v' \in \mathbb{Z}^m$ such that $v' \ne v$ and $Mv \equiv Mv' \bmod q$.

Setting $u = v - v'$ yields a nonzero vector in $\mathbb{Z}^m$ satisfying $|u| \le m$ and $Mu \equiv 0 \bmod q$. This vector u is the output of the algorithm A on input M. Applying Theorem 49 to A yields the $\mathrm{BPP}^{R_{\mathrm{KT}}}$ algorithm for SIVP.

The reductions for SBP, Unique-SVP, and CVP follow from the one for SIVP by arguments given in [CN97, Cai01]. The results for LSVP and CRP are obtained in a similar way based on the variants of the candidate one-way function f and the worst-case to average-case connection corresponding to Theorem 49 presented in [Mic02]. □

4.3 R_{KT} versus NP

Given our inability to prove that R_{KT} is coNP-complete, one may wonder whether R_{KT} is in NP. If it is, then this would provide a dense combinatorial property in NP that is useful against P/poly, contrary to a conjecture of Rudich [Rud97]. We can also show the following.

Theorem 50 *If R_{KT} is in* NP, *then* MA = NP.

Proof. It is shown in [IKW02] that if an NP machine can, on input of length n, find the truth table of a function of size $n^{O(1)}$ with large circuit complexity, then MA = NP. Certainly this is easy if R_{KT} is in NP. □

This observation (similar to ones in [IKW02]) can not be taken as evidence that $R_{\mathrm{KT}} \notin$ NP, since many conjecture that MA is equal to NP. However, it does show that proving R_{KT} in NP would require nonrelativizing proof techniques.

5 Open Problems

In this paper, we present KT as a notion of time-bounded Kolmogorov complexity with close connections to circuit complexity. In so doing, we expose connections between circuit complexity and traditional Kolmogorov complexity and Levin's measure Kt. KT-complexity is also useful as a tool for summarizing some recent progress in the field of derandomization, and for describing the theory of natural proofs from the standpoint of Kolmogorov complexity. For an exposition along these lines, see [All01].

The measures KT, KS, and Kt provide natural and interesting examples of apparently intractable problems in NP, PSPACE, and EXP that are not complete under the more familiar notions of reducibility and hence constitute a fundamentally new class of complete problems. It is worth pointing out that variants of these sets (such as $\{x : \mathrm{Kt}(x) \geq |x|/3\}$) appear to be incomparable to our standard $R_{\mathrm{Kt}} = \{x : \mathrm{Kt}(x) \geq |x|/2\}$ under $\leq^{\mathrm{p}}_{\mathrm{m}}$ reductions. As another example of this phenomenon, we are unable to present any reduction between $\{x : \mathrm{KT}_U(x) \geq |x|/2\}$ and $\{x : \mathrm{KT}_{U'}(x) \geq |x|/2\}$, where U and U' are two universal machines. This deserves further investigation.

Corollaries 39 and 36 combine to show that EXP = ZPP if and only if $R_{\mathrm{Kt}} \in$ ZPP. This is because $R_{\mathrm{Kt}} \in$ EXP and if $R_{\mathrm{Kt}} \in$ ZPP then EXP = PSPACE = $\mathrm{ZPP}^{R_{\mathrm{Kt}}}$. This suggests that one might be able to show that R_{Kt} is complete under ZPP reductions – but we have not been able to show this.

Can one settle (one way or the other) the question of whether or not R_{Kt} is complete for EXP under $\leq^{\mathrm{p}}_{\mathrm{T}}$ reductions? For instance, is it possible to construct an element of R_{Kt} in $\mathrm{P}^{R_{\mathrm{Kt}}}$ (which would suffice to prove completeness, as in the proof of Theorem 41)? Or is R_{Kt} more similar to the Buhrman-Mayordomo set R^t (which is provably *not* complete) in this regard?

Is there some sense in which KT complexity yields NP-complete sets? For instance, what can one say about $\{(x, y, i) : \mathrm{KT}(x|y) \leq i\}$? The result of [VV83] is intriguing in this light; in [VV83] Vazirani and Vazirani presented a language in NP that is complete under probabilistic reductions that is not known to be complete under deterministic reductions. Their problem superficially seems to be related to time-bounded Kolmogorov complexity.

Other intriguing open questions are: Is PSPACE $\subseteq \mathrm{P}^{R_{\mathrm{KS}}}$? Is the EXP-complete set R_{Kt} in P? Is it in $\mathrm{AC}^0[\oplus]$? We know of no fundamental obstacles that lie in the way of a direct proof that

R_{Kt} is not in P, although it was pointed out to us by Rahul Santhanam (personal communication) that EXP = ZPP if and only if there is a set in P of polynomial density containing only strings of Kt-complexity $\geq n^{\epsilon}$. The backward implication follows via the same argument that we used to show that EXP = ZPP if and only if $R_{\mathrm{Kt}} \in$ ZPP. For the other direction, assume that EXP = ZPP, and let B be a set in EXP of polynomial density such that neither B nor its complement has an infinite subset in DTime(2^n) [GHS91]. Let A be the set in P consisting of (string,witness) pairs for the ZPP machine accepting B. If A contains infinitely many strings of low Kt complexity, then B contains infinitely many instances where membership can be decided in time 2^n, contrary to our choice of B.

The containment of PSPACE in $\mathrm{P}^{R_{\mathrm{C}}}$ raises the question of whether it might be possible to obtain characterizations of complexity classes in terms of efficient reductions to R_{C}. This is studied in more detail in [ABK04] (where a nontrivial characterization of P having this flavor is presented).

Acknowledgments

We acknowledge illuminating discussions with Lance Fortnow, as well as helpful conversations of the first author with Paul Beame, Henry Cohn, Chris Umans, and Ramarathnam Venkatesan during a visit to Microsoft Research. We thank Jin-Yi Cai and Daniele Micciancio for clarifying some issues regarding lattice problems. We also acknowledge fruitful discussions with Steven Rudich, Valentine Kabanets, Manindra Agrawal, Troy Lee, Neil Jones, Rahul Santhanam, Kolya Vereshchagin, John Hitchcock, and Pavan Aduri. Finally, we thank the anonymous referees for their suggestions.

References

[AAI+01] M. Agrawal, E. Allender, R. Impagliazzo, R. Pitassi, and S. Rudich. Reducing the complexity of reductions. *Computational Complexity*, 10:117–138, 2001.

[ABK+02] E. Allender, H. Buhrman, M. Koucký, D. van Melkebeek, and D. Ronneburger. Power from random strings. In *Proceedings of the 43th Annual Symposium on Foundations of Computer Science*, pages 669–678, 2002.

[ABK04] E. Allender, H. Buhrman, and M. Koucký. What can be efficiently reduced to the K-random strings? *Annals of Pure and Applied Logic*, 2004. To appear; a preliminary version appeared in Proc. Symposium on Theoretical Aspects of Computer Science, 2004, Lecture Notes in Computer Science vol. 2996.

[AFvM01] L. Antunes, L. Fortnow, and D. van Melkebeek. Computational depth. In *IEEE Conference on Computational Complexity*, pages 266–273, 2001.

[AG91] E. Allender and V. Gore. Rudimentary reductions revisited. *Information Processing Letters*, 40:89–95, 1991.

[Agr01] M. Agrawal. The first-order isomorphism theorem. In *Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS)*, volume 2245 of *Lecture Notes in Computer Science*, pages 70–82, 2001.

[Ajt96] M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *ACM Symposium on Theory of Computing (STOC)*, pages 99–108, 1996.

[Ajt04] M. Ajtai. Generating hard instances of lattice problems. In J. Krajíček, editor, *Complexity of Computations and Proofs*, volume 13 of *Quaderni di Matematica*, pages 1–32. Seconda Università di Napoli, 2004. Preliminary version appeared as [Ajt96].

[AKRR03] E. Allender, M. Koucký, D. Ronneburger, and S. Roy. Derandomization and distinguishing complexity. In *Proceedings of the 18th IEEE Conference on Computational Complexity*, pages 209–220, 2003.

[AKS04] M. Agrawal, N. Kayal, and N. Saxena. PRIMES is in P. *Annals of Mathematics*, 160(2):781–793, 2004.

[All01] E. Allender. When worlds collide: Derandomization, lower bounds, and Kolmogorov complexity. In *Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS)*, volume 2245 of *Lecture Notes in Computer Science*, pages 1–15, 2001.

[All05] E. Allender. NL-printable sets and nondeterministic Kolmogorov complexity. *Theoretical Computer Science*, 2005. To appear; a preliminary version appeared Proc. 10th Workshop on Logic, Language, Information and Computation (WoLLIC'2003).

[BAJ00] A. Ben-Amram and N. Jones. Computational complexity via programming languages: Constant factors do matter. *Acta Informatica*, 37:83–120, 2000.

[Ber78] P. Berman. Relationship between density and deterministic complexity of NP-complete languages. In *ICALP*, volume 62 of *Lecture Notes in Computer Science*, pages 63–71, 1978.

[BFL91] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.

[BFNV05] H. Buhrman, L. Fortnow, I. Newman, and N. Vereshchagin. Increasing Kolmogorov complexity. In *Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 3404 of *Lecture Notes in Computer Science*, pages 412–421, 2005.

[BFNW93] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3:307–318, 1993.

[BIS90] D. Mix Barrington, N. Immerman, and H. Straubing. On uniformity within NC^1. *Journal of Computer and System Sciences*, 41(3):274–306, December 1990.

[BM84] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudorandom bits. *SIAM Journal on Computing*, 13:850–864, 1984.

[BM97] H. Buhrman and E. Mayordomo. An excursion to the Kolmogorov random strings. *Journal of Computer and System Sciences*, 54:393–399, 1997.

[BT01] H. Buhrman and L. Torenvliet. Randomness is hard. *SIAM Journal on Computing*, 30:1485–1501, 2001.

[Cai99] J.-Y. Cai. Some recent progress on the complexity of lattice problems. In *IEEE Conference on Computational Complexity*, pages 158–179, 1999.

[Cai01] J.-Y. Cai. On the average-case hardness of CVP. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 308–319, 2001.

[CN97] J.-Y. Cai and A. Nerurkar. An improved worst-case to average-case connection for lattice problems. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 468–477, 1997.

[FF93] J. Feigenbaum and L. Fortnow. On the random-self-reducibility of complete sets. *SIAM Journal on Computing*, 22:994–1005, 1993.

[Für82] M. Fürer. The tight deterministic time hierarchy. In *ACM Symposium on Theory of Computing (STOC)*, pages 8–16, 1982.

[Für84] M. Fürer. Data structures for distributed counting. *Journal of Computer and System Sciences*, 28(2):231–243, 1984.

[GGM86] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, 1986.

[GHS91] J. Geske, D. Huynh, and J. Seiferas. A note on almost-everywhere-complex sets and separating deterministic-time-complexity classes. *Information and Computation*, 92:97–104, 1991.

[HILL99] J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28:1364–1396, 1999.

[HS66] F. Hennie and R. Stearns. Two-tape simulation of multitape turing machines. *Journal of the ACM*, 13:533–546, 1966.

[IKW02] R. Impagliazzo, V. Kabanets, and A. Wigderson. In search of an easy witness: Exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65:672–694, 2002.

[IW97] R. Impagliazzo and A. Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In *ACM Symposium on Theory of Computing (STOC) '97*, pages 220–229, 1997.

[IW01] R. Impagliazzo and A. Wigderson. Randomness vs. time: de-randomization under a uniform assumption. *Journal of Computer and System Sciences*, 63:672–688, 2001.

[Kab01] V. Kabanets. Easiness assumptions and hardness tests: Trading time for zero error. *Journal of Computer and System Sciences*, 63(2):236–252, 2001.

[KC00] V. Kabanets and J.-Y. Cai. Circuit minimization problem. In *ACM Symposium on Theory of Computing (STOC)*, pages 73–79, 2000.

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

[Kou03] M. Koucký. *On Traversal Sequences, Exploration Sequences, and Completeness of Kolmogorov Random Strings.* PhD thesis, Rutgers University, 2003.

[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, 1996.

[KvM02] A. Klivans and D. van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM Journal on Computing*, 31:1501–1526, 2002.

[Lad75] R. Ladner. On the structure of polynomial time reducibility. *Journal of the ACM*, 22:155–171, 1975.

[Lev84] L. Levin. Randomness conservation inequalities; information and independence in mathematical theories. *Information and Control*, 61:15–37, 1984.

[LL76] R. Ladner and N. Lynch. Relativization of questions about log space reducibility. *Mathematical Systems Theory*, 10:19–32, 1976.

[Lon86] L. Longpré. *Resource Bounded Kolmogorov Complexity, A Link between Computational Complexity and Information Theory.* PhD thesis, Cornell University, 1986.

[LV93] M. Li and P. Vitanyi. *Introduction to Kolmogorov Complexity and its Applications.* Springer, 1993.

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

[Mic02] D. Micciancio. Improved cryptographic hash functions with worst-case / average-case connection. In *ACM Symposium on Theory of Computing (STOC)*, pages 609–618, 2002.

[Nis91] N. Nisan. Pseudorandom bits for constant depth circuits. *Combinatorica*, 11:63–70, 1991.

[NW94] N. Nisan and A. Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, October 1994.

[Rab79] M. Rabin. Digitalized signatures and public-key functions as intractible as factorization. MIT Tech. Report TR-212, 1979.

[Ron04] D. Ronneburger. *Kolmogorov Complexity and Derandomization.* PhD thesis, Rutgers University, 2004.

[RR97] A. Razborov and S. Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55:24–35, 1997.

[Rud97] S. Rudich. Super-bits, demi-bits, and $\tilde{\mathrm{NP}}$/qpoly-natural proofs. In *RANDOM*, volume 1269 of *Lecture Notes in Computer Science*, 1997.

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

[STV01] M. Sudan, L. Trevisan, and S. Vadhan. Pseudorandom generators without the XOR lemma. *Journal of Computer and System Sciences*, 62:236–266, 2001.

[Tre01] L. Trevisan. Construction of extractors using pseudo-random generators. *Journal of the ACM*, 48(4):860–879, 2001.

[TV02] L. Trevisan and S. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. In *IEEE Conference on Computational Complexity*, pages 129–138, 2002.

[Vio04] E. Viola. The complexity of constructing pseudorandom generators from hard functions. *Computational Complexity*, 13:147–188, 2004.

[Vol99] H. Vollmer. *Introduction to Circuit Complexity.* Springer, 1999.

[VV83] U. Vazirani and V. Vazirani. A natural encoding scheme proved probabilistic polynomial complete. *Theoretical Computer Science*, 24:291–300, 1983.

[Wat87] O. Watanabe. A comparison of polynomial time completeness notions. *Theoretical Computer Science*, 54:249–265, 1987.

[WT92] O. Watanabe and S. Tang. On polynomial-time Turing and many-one completeness in PSPACE. *Theoretical Computer Science*, 97:199–215, 1992.

[Yao82] A. Yao. Theory and applications of trapdoor functions. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 80–91, 1982.