

# Applications of Time-Bounded Kolmogorov Complexity in Complexity Theory

*Eric Allender\**

Department of Computer Science  
Rutgers University  
New Brunswick, NJ 08903, USA  
allender@cs.rutgers.edu

**Abstract.** This paper presents one method of using time-bounded Kolmogorov complexity as a measure of the complexity of sets, and outlines a number of applications of this approach to different questions in complexity theory. Connections will be drawn among the following topics: NE predicates, ranking functions, pseudorandom generators, and hierarchy theorems in circuit complexity.

## 1 Introduction

Complexity theory provides a setting in which one can associate to any recursive set  $L$  a function  $t_L$  on the natural numbers, and with justification claim that  $t_L$  is a measure of the complexity of  $L$ ; namely  $L$  can be accepted by exactly those machines that run in time  $\Omega(t_L(n))$ . In this paper, we will consider a means of using time-bounded Kolmogorov complexity to define a function  $K_L$ , that measures a different aspect of the complexity of  $L$ . We will argue that this is a useful measure by presenting a number of applications of this measure to questions in complexity theory.

### 1.1 Complexity of Strings

Before going any further, it is necessary to define the sort of time-bounded Kolmogorov complexity that we will be considering. Many alternate approaches exist for adding a time-complexity component to Kolmogorov complexity. Sipser [Sip83] and Ko [Ko86] proposed essentially identical definitions, allowing one to define, for each function  $f$ , a  $f(n)$  time-bounded Kolmogorov complexity measure  $K^f$ , where  $K^f(x)$  is the length of the shortest description of  $x$  from which  $x$  can be produced in  $f(|x|)$  steps. A related (and much more influential) definition due to Hartmanis [Har83] yields sets of the form  $K[g(n), G(n)]$ , consisting of all strings  $x$  that can be produced from a description of length  $g(|x|)$  in time  $G(|x|)$ . Pointers to other approaches to time-bounded Kolmogorov complexity may be found in [All89a, LV90].

---

\* Supported in part by the National Science Foundation under Grant CCR-9000045.

The variants of time-bounded Kolmogorov complexity mentioned in the preceding paragraph all suffer from certain drawbacks. For example, the definitions of Ko and Sipser provide a family of measures  $K^f$  but offer no guidance in selecting any one function  $f$  as the preferred choice when defining *the* time-bounded Kolmogorov complexity of a string  $x$ . Additionally, for any given  $f(n) \gg n$ , the measure  $K^f$  assigns the same complexity to a string  $x$ , regardless of whether it can be built from a short description in linear time or requires time  $f(|x|)$ , and thus some important distinctions can not be made. The definition of Hartmanis does allow many fine distinctions to be made, but does not provide a *function* measuring the complexity of a string  $x$ ; the time and length parameters are not combined in any way.

Thus we turn to another version of time-bounded Kolmogorov complexity: a definition due to Levin [Lev84] (see also [Lev73]).

**Definition 1.** [Lev84] For any strings  $x$  and  $z$ , and for any Turing machine  $M_v$ , define  $\text{Kt}_v(x|z)$  to be

$$\min\{|y| + \log t : M_v(y, z) = x \text{ in at most } t \text{ steps}\}.$$

$\text{Kt}_v(x)$  is defined to be  $\text{Kt}_v(x|\lambda)$ , where  $\lambda$  denotes the empty string. Via a standard argument, one can show the existence of a “universal” Turing machine<sup>2</sup>  $M_u$  such that, for all  $v$  there exists a  $c$  such that for all  $x$   $\text{Kt}_v(x) \geq \text{Kt}_u(x) + c + \log \log \text{Kt}_v(x)$ . Choose some such universal Turing machine, and define  $\text{Kt}(x|z)$  to be  $\text{Kt}_u(x|z)$ , and  $\text{Kt}(x) = \text{Kt}_u(x)$ .

It is clear that Levin’s definition overcomes the objections raised above. However, it may be less clear that Levin’s definition is the appropriate definition – or even a reasonable one.

What is the motivation for defining the complexity of  $x$  to be the minimum of the sum of the description length and the log of the time required to build  $x$  from that description? The answer to this question is that this is precisely the combination of time and description length that is most useful in the study of problems such as the P versus NP question. Consider the problem of finding a satisfying assignment for a formula  $\phi$  with  $n$  variables. When searching through all the  $2^n$  possible assignments to the variables of  $\phi$ , what is the optimal search strategy that will lead to a satisfying assignment as quickly as any? The answer, as noted by Levin [Lev73] is to consider each string  $z \in \Sigma^n$  in order of increasing  $\text{Kt}(z|\phi)$ . Levin also used this approach to provide bounds on “speed-up” (in the sense of Blum’s speed-up theorem [Blu67]) possible for the problem of inverting a polynomial-time computable permutation.

Levin’s  $\text{Kt}$  function is clearly closely related to the generalized Kolmogorov complexity sets defined by Hartmanis:

---

<sup>2</sup> Levin actually defines  $\text{Kt}$ -complexity using a different model of computation, allowing the  $\log \log$  term to be eliminated; for simplicity, we will stick to the Turing machine model of computation in this paper, as the  $\log \log$  terms are insignificant for our purposes.

**Proposition 2.**  $Kt(x) \leq s(|x|) \Rightarrow x \in K[s(n), 2^{s(n)}] \Rightarrow Kt(x) \leq 2s(|x|)$ .

As mentioned above, Hartmanis' formulation has the advantage that one is able to discuss separately the *size* of a string's description and the *time* required to build the string; thus some finer distinctions can be made. However, one of our goals in this section is to define a measure of the complexity of a language, and for this purpose Levin's  $Kt$  function combines the time and size components in the most appropriate fashion.

## 1.2 Complexity of Languages

Now that we have settled on a measure of the time-bounded Kolmogorov complexity of strings, let us consider how to define a complexity measure for languages.

Perhaps the most obvious way to use Kolmogorov complexity to measure the complexity of a language  $L$  is to consider the characteristic sequence of  $L$ : the sequence  $a_1, a_2, \dots$  where  $a_i$  is zero or one, according to whether or not  $x_i \in L$ , where  $x_1, x_2, \dots$  is an enumeration of  $\Sigma^*$ . Investigations of this sort may be found in [Ko86, Huy85, Huy86, BDG87, MS90, Lut91]. For example, in [BDG87], it was shown that PSPACE/poly is the class of all languages  $L$  such that each finite prefix of the characteristic sequence of  $L$  has small space-bounded Kolmogorov complexity.

It is often useful, however, to consider the complexity of the individual strings in a language  $L$ , as opposed to the characteristic sequence of  $L$ . This leads us to the following definitions [All89a].

**Definition 3.** Let  $L \subseteq \{0, 1\}^*$ . Then we define:

- $K_L(n) = \min\{Kt(x) : x \in L^n\}$
- $K^L(n) = \max\{Kt(x) : x \in L^n\}$

If there are no strings of length  $n$  in  $L$ , then  $K_L(n)$  and  $K^L(n)$  are both undefined. When we consider the rate of growth of functions of the form  $K_L(n)$ , the undefined values are not taken into consideration. Thus, for example, we say  $K_L(n) = O(\log n)$  if there is some constant  $c$  such that, for all large  $n$ , if  $K_L(n)$  is defined, then  $K_L(n) < c \log n$ . Similarly,  $K_L(n) \neq \omega(s(n))$  if there is some constant  $c$  such that, for infinitely many  $n$ ,  $K_L(n)$  is defined and  $K_L(n) \leq cs(n)$ .

If, for some language  $L$ , the function  $K^L$  has a slow rate of growth, then this says that *all* of the strings in  $L$  have small time-bounded Kolmogorov complexity. In particular,  $K^L(n) = O(\log n)$  if and only if  $L \subseteq K[k \log n, n^k]$  for some  $k$ . Sets with this property have been studied extensively in recent years; the interested reader will find material concerning these sets, along with pointers to the relevant literature, in the survey article by R. V. Book [Boo92]. Because of this, we will not dwell on the  $K^L$  measure, and will focus instead on the  $K_L$  measure throughout the rest of this paper.

It is immediate from the definition that for any language  $L$ ,  $K_L(n) \leq n + \log n + O(1)$ , and  $K_L(n) = \Omega(\log n)$ . The question of how quickly  $K_L(n)$  may

grow, when  $L$  is a set in P, turns out to have many connections to a variety of questions in complexity theory, and the rest of this paper is devoted to exploring some of those connections, beginning with questions concerning deterministic and nondeterministic exponential time.

## 2 NE Predicates

Let the complexity classes  $\text{DTIME}(2^{O(n)})$  and  $\text{NTIME}(2^{O(n)})$  be denoted by E and NE, respectively. These are simply the exponential-time analogs of P and NP, and the E=NE question is generally considered to be of essentially the same level of difficulty as the famous P=NP question.

Of course, much of the motivation for the complexity class NP comes from so-called “search” problems: for example, the problem of searching for a Hamiltonian path in a graph  $G$ , or the problem of producing a satisfying assignment for a Boolean formula  $\phi$  if one exists. In contrast to search problems, language recognition problems in NP are questions about the *existence* of solutions. (E.g.,  $\phi \in \text{SAT}$  iff a satisfying assignment for  $\phi$  *exists*.) In practical applications, it is of little use to know merely that a Hamiltonian path of a given weight *exists* in a graph – it is much more important to have the path itself. The P=NP question itself is usually phrased in terms of language recognition instead of search problems, but it is a well-known fact that P=NP if and only if all of the related search problems are solvable in polynomial time.

An analogous notion of “search problem” may be defined for NE:

**Definition 4.** An *NE-predicate* is a relation  $R$  defined by an exponential-time nondeterministic Turing machine  $M$ ;  $R(x, y)$  is true iff  $y$  encodes an accepting computation of  $M$  on input  $x$ .  $R$  is *solvable in time  $T$*  if there is a deterministic Turing machine running in time  $T$  that, on input  $x$ , finds a string  $y$  such that  $R(x, y)$  holds, if any such  $y$  exists.

Stated another way,  $R$  is solvable if there is a routine that, for all  $x$ , can find a witness for  $x$  if a witness exists. Conversely,  $R$  is not solvable in time  $t$  if each routine running in time  $t$  fails to find witnesses for infinitely many  $x$  that have witnesses. Note however that non-solvability of  $R$  says nothing about the frequency with which “hard” inputs are encountered. Note that the strongest statement about non-solvability of an NE predicate that one could make would be to say that *all* large inputs are “hard” in this sense. This leads to the following definition, which generalizes the classical notion of immunity.

**Definition 5.** An NE predicate  $R$  is *immune* with respect to time  $t(n)$  if (1) the set  $\{x : \exists y R(x, y)\}$  is infinite, and (2) for all  $f$  computable in time  $t(n)$ , the set  $\{x : R(x, f(x))\}$  is finite.

The connections between NE predicates and Kolmogorov complexity were first drawn in [AW90]; the following theorem is a slight generalization of the results presented there. In short, it says that there are hard NE predicates if and only if there are sets  $L$  in P such that  $K_L(n)$  grows quickly.

**Theorem 6.** (a) *Every NE predicate is solvable in time  $t(2^{O(n)})$  iff for every set  $L$  in  $P$ ,  $K_L(n) = O(\log t(n^{O(1)}))$*   
 (b) *No NE predicate is immune with respect to time  $t(2^{O(n)})$  iff for every set  $L$  in  $P$ ,  $K_L(n) \neq \omega(\log t(n^{O(1)}))$*

*Proof.* We will sketch a proof of the first equivalence; the second equivalence is proved in a very similar manner. For the forward direction, assume that NE predicates can be solved in the stated time bound, and let  $L$  be a set in  $P$ . Let  $R$  be the relation given by  $R(m, x) \Leftrightarrow x \in L^m$ . (Here, we assume the standard binary representation of numbers, and identify a number with its binary representation.) Then  $R$  is an NE predicate and there is a function  $f$  computable in time  $t(2^{cn})$  that solves  $R$ , for some constant  $c$ . Let  $s$  be a description of a machine computing  $f$ . Now note that if  $L^m$  is nonempty, then  $f(m) \in L^m$ , and  $\text{Kt}(f(m)) \leq |s| + |m| + \log t(2^{|m|}) = O(1) + \log |f(m)| + \log t(|f(m)|^{O(1)})$ , which establishes the forward direction.

For the converse, assume the given bounds on the rate of growth  $K_L$  for sets  $L$  in  $P$ , and let  $R$  be any NE predicate defined by a machine  $M$  running in time  $2^{cn}$ . Let  $L$  be the set of all strings  $z$  of length  $m^c$  for some  $m$  such that there is some prefix  $w$  of  $z$  such that  $R(m, w)$ .  $L$  is in  $P$ , and by assumption  $K_L(n) \leq d \log t(n^d)$  for some  $d$ . Consider the following routine for solving the NE predicate  $R$ : On input  $m$ , for each string  $y$  of length at most  $d \log t(m^d)$ , run  $M_u$  on input  $y$  for  $t(m^d)^d$  steps and see if the output produced has a prefix encoding an accepting computation of  $M$  on input  $m$ . Output the first accepting computation found in this way, if any is found. It is easily verified that this routine solves  $R$  within the claimed time bound.  $\square$

**Corollary 7.** (a) *Every NE predicate is solvable in exponential time iff  $K_L(n) = O(\log n)$  for all  $L$  in  $P$ .*  
 (b) *No NE predicate is immune with respect to exponential time iff  $K_L(n) \neq \omega(\log n)$  for all  $L$  in  $P$ .*

An essentially identical proof shows that if there is an NE predicate that cannot be solved in time  $2^{2^{n-1}}$ , then there is a set  $L$  in  $P$  with  $K_L(n) \geq n/5$ . We thus see that the most common conjectures concerning the difficulty of sets in NE have as a consequence that there are some sets  $L$  in  $P$  with rather high Kolmogorov complexity, as measured by the function  $K_L$ .

From Theorem 6, we see that the question of whether or not a set  $L$  in  $P$  can have nontrivial growth rate is very closely related to the  $E=NE$  question. It is natural to ask if it is in fact *equivalent* to  $E=NE$ . Note that if every NE predicate is solvable in exponential time, then  $E=NE$  is a trivial consequence; does the converse hold?

This question was explicitly raised in [AW90] as a result of an investigation using Kolmogorov complexity as a tool for answering certain questions concerning classes of sets equivalent to tally sets<sup>3</sup> under varying notions of reducibility.

<sup>3</sup> A set is a *tally* set if it is a subset of  $0^*$ .

(See [Boo92] for a discussion.) The question was essentially answered by Impagliazzo and Tardos [IT89]; they present an oracle relative to which  $E=NE$  but there are NE predicates that cannot be solved in exponential time. Thus the equivalence of the language recognition and witness-finding problems as it relates to the P versus NP problem does not generalize to larger time bounds.

### 3 Related Notions

The questions of whether or not  $K_L(n) = O(\log n)$  or  $K_L(n) \neq \omega(\log n)$  for all  $L$  in P have been asked many times by different researchers studying apparently unrelated topics. In this section, we will gather some of these diverse results together.

#### 3.1 P-Printability

A set  $L$  is *P-printable* if there is an algorithm that can list all of the elements of  $L^n$  in time polynomial in  $n$ . An immediate consequence of this definition is that all P-printable sets are sparse and are in P. However, it is suspected that there are sparse sets in P that are not P-printable. P-printable sets were defined in [HY84] and have been studied in many papers; for further information see [Boo92]. It was shown in [AR88] (see also [HH88]) that  $L$  is P-printable iff  $L$  is in P and  $K^L(n) = O(\log n)$ .

Many of the papers that consider P-printable sets ask if every infinite set in P has an infinite P-printable subset. It is an easy observation that a set  $L$  in P has an infinite P-printable subset iff  $K_L(n) \neq \omega(\log n)$ . Furthermore, it was observed by Russo (see [AR88]) that sets in NP are similar to sets in P in this regard: every set in P has an infinite P-printable subset iff every set in NP has an infinite P-printable subset. Rephrasing Russo's observation in terms of Kolmogorov complexity, we get:

**Theorem 8.** (a)  $K_L(n) \neq \omega(\log n)$  for all  $L$  in P iff  $K_L(n) \neq \omega(\log n)$  for all  $L$  in NP  
 (b)  $K_L(n) = O(\log n)$  for all  $L$  in P iff  $K_L(n) = O(\log n)$  for all  $L$  in NP

#### 3.2 Ranking

The property of having an infinite P-printable subset was called “tangibility” in [HR90]. It was studied there as a very weak notion related to a concept known as *ranking*. As there are other connections with ranking to explore, let us consider ranking more closely.

**Definition 9.** Let  $L$  be a language. The function defined by  $rank_L(x) = \|\{y \in L : y \leq x\}\|$  is known as the *ranking function* for  $L$ . (Here,  $\leq$  denotes lexicographic order.)

If  $L$  has an easily-computed ranking function, then there is an efficiently-computable bijection from  $L$  onto the natural numbers; each element of  $L$  is mapped to its rank in  $L$ . This bijection may be thought of as mapping each element of  $L$  to a “compressed” representation. If  $L$  contains only a small fraction of the words of each length and has an easily-computed ranking function this clearly places bounds on the  $K_L$ -complexity of any string in  $L$ . Ranking functions were first studied in [GS91] in connection with data compression. Other material on ranking may be found in [All85, BGS91, BGM90, Huy90a, HR90]. A number of classes of sets (including the unambiguous context-free languages [GS91]) have been shown to have easy ranking functions.

If  $L$  has an easy ranking function, then a simple binary search procedure enables one to quickly locate the lexicographically least element of  $L^n$ . It thus follows that any such  $L$  has  $K_L(n) = O(\log n)$ . Thus all the classes of sets shown in [All85, BGS91, BGM90, Huy90a, GS91] to have easy ranking functions are also classes of sets with low  $K_L$  complexity.

It is worth noting, however, that the class of sets in  $P$  with low  $K_L$  complexity is somewhat larger than the class of sets with easy ranking functions. For example, it is not hard to see that if  $L$  is a context-free language, then  $K_L(n) = O(\log n)$ . (The proof is quite similar to the proof of Theorem 4 in [AR88]; see also [All85, Huy90b].) However, it was shown in [Huy90a] that if  $P \neq PP$  there are context-free languages that have hard ranking functions. As another example, it was shown in [Huy90a] that if every set in  $NTIME(\log n)$  has an easy ranking function,<sup>4</sup> then  $P = PP$ , although Gore has observed [Gor90] that  $K_L(n) = O(\log n)$  for all  $L \in NTIME(\log n)$ . On the other hand, Huynh’s techniques [Huy90a] can easily be used to prove that there are sets  $S \in coNTIME(\log n)$  with  $K_S \neq O(\log n)$  unless  $K_L(n) = O(\log n)$  for all  $L \in P$  (which is to say, unless all NE predicates are solvable in exponential time).

### 3.3 Invertibility

Note that finding an element of  $L^n$  is roughly the same as computing a sort of inverse of the characteristic function of  $L$ ,  $\chi_L$ ; the aim is to find a string in a certain range that is in  $\chi_L^{-1}(1)$ . In [All85] a number of classes of automata are exhibited that compute only easy-to-invert functions.<sup>5</sup> If  $L$  is a language accepted by any machine in one of these classes, then  $K_L(n) = O(\log n)$ .

The connection to invertible functions was stated more precisely in [AW90]. It is shown there that  $K_L(n) = O(\log n)$  for all  $L \in P$  iff every honest function  $f : \Sigma^* \rightarrow 0^*$  computable in polynomial time is weakly invertible.<sup>6</sup>

<sup>4</sup> In order to consider sublinear running times, the Turing machine model considered here has an “index tape” allowing the machine to access a given input position in unit time. This model is commonly used when using the Turing machine formalism to characterize circuit complexity classes.

<sup>5</sup> In a related result, some of these same classes of machines were shown in [KGY89] to be unable to compute pseudorandom generators.

<sup>6</sup> A function  $f$  is *honest* if  $|f(x)|$  is polynomially related to  $|x|$ ;  $f$  is *weakly invertible* if there is a function  $g$  computable in polynomial time such that  $f(g(x)) = x$  for all

### 3.4 Generation

In work originally motivated by the problem of generating test data for heuristic testing and evaluation, Sanchis and Fulk [SF90] defined the notion of a *polynomial-time constructor (PTC)* for a language  $L$ , which is a routine running in polynomial time that on input  $1^n$  either produces an element of  $L^n$  or announces that  $L^n = \emptyset$ . Clearly, if  $L$  has a PTC, then  $K_L(n) = O(\log n)$ , and if  $L$  is in P, then  $L$  has a PTC iff  $K_L(n) = O(\log n)$ . (Additional work dealing with constructors is reported in [Huy90b].)

It is noted in [SF90] that most “natural” sets of interest in practical situations are easily seen to have PTCs. Thus sets  $L$  in P such that  $K_L$  grows rapidly seem to be somewhat “unnatural” – but note that such sets must exist, unless  $E=NE$ .

## 4 Pseudorandom Generators

A pseudorandom generator is an efficient algorithm that takes a short input (the *random seed*) and produces a long pseudorandom output. A pseudorandom generator is *secure* if the output produced passes all “feasible” statistical tests for randomness. Pseudorandom generators are the object of much study in the theory of cryptography; an excellent introduction to the theory of pseudorandom generators may be found in [BH88].

There are many different ways of formalizing the hypothesis “secure pseudorandom generators exist”, depending on the particular notion of statistical test being used, and depending on the desired degree of “security”. In this overview, we will try to present material on an intuitive level; the reader is invited to consult the cited references for more precise definitions.

For the purposes of this section we will use the definitions of [Yao82]. A statistical test is a language  $L$  in P/poly (that is, a set accepted by a (possibly nonuniform) family of polynomial-size circuits). For each input length  $n$ , the probability that  $L$  contains a random input of length  $n$  is simply  $\|L^n\|/2^n$ . If a pseudorandom generator  $f$  takes inputs of length  $n$  and produces output of length  $p(n)$ , then the probability that  $L$  contains a pseudorandom input of length  $p(n)$  is  $\|\{y \in \Sigma^{p(n)} : f(y) \in L\}\|/2^n$ . The generator  $f$  will be said to pass the statistical test  $L$  if for all polynomials  $q$  and for all large  $n$ , the probabilities that  $L$  contains random and pseudorandom strings of length  $p(n)$  differ by at most  $1/q(n)$ . If  $f$  passes all statistical tests in P/poly, then  $f$  is said to be *secure*. That is,  $f$  is secure if the pseudorandom output it produces “looks random” to all tests in P/poly.

In [All89a], it was shown that most ways of formalizing the hypothesis that secure pseudorandom generators exist have as a consequence that  $K_L(n)$  grows slowly for all dense sets  $L$  in P.<sup>7</sup> For the purposes of this section, the following

---

$x$  in the image of  $f$ .

<sup>7</sup> In the following, the *density* of a set  $L$ , denoted  $d_L(n)$ , is the function given by  $d_L(n) = \|\{L^n\}\|/2^n$ . A set  $L$  is said to be *dense* if for some  $k$  and all large  $n$ , if  $L^n \neq \emptyset$ , then  $d_L(n) \geq 1/n^k$ . That is,  $L$  is dense if it contains many strings of length  $n$ , if it contains any at all.

result is illustrative.

**Theorem 10.** [All89a] *If there are pseudorandom generators that are secure against  $P/poly$  statistical tests, then  $K_L(n) = O(n^\epsilon)$  for all dense sets  $L$  in  $P/poly$ , and for all  $\epsilon > 0$ .*

Variants of Theorem 10 were used in [All89a] to show new inclusion relations among complexity classes, under various assumptions about the security of pseudorandom generators.

It is known that pseudorandom generators exist if and only if one-way functions exist that are hard to invert over a significant fraction of their range [ILL89, Hås90]. Thus the existence of this sort of one-way function implies that  $K_L$  cannot grow too quickly for any dense set  $L$  in  $P$ . On the other hand, we saw in the preceding section that the solvability of NE predicates was also equivalent to the existence of a certain sort of one-way function, and the existence of this sort of one-way function implies that  $K_L$  must grow quickly for some sets  $L$  in  $P$ . Thus there seems to be some sort of trade-off concerning what notions of one-way-ness are mutually compatible; this situation is still only poorly understood.

Taken together, Theorems 6 and 10 motivate the question of whether or not there is any connection between the density of a set  $L$  in  $P$  and the rate of growth of  $K_L(n)$ . That is,

- NE contains hard sets  $\Rightarrow K_L(n)$  grows quickly for some set  $L$  in  $P$ .
- Secure pseudorandom generators exist  $\Rightarrow K_L(n)$  grows slowly for all dense sets  $L$  in  $P$ .

Since the left hand sides of these implications are often conjectured to hold, it follows that it is conjectured that  $K_L(n)$  can achieve a faster growth rate if  $L$  is sparse than if  $L$  is dense.

Apart from observations such as these, there is little to guide one's intuition in questions concerning the  $K_L$  complexity of sets  $L$  in  $P$ . In the following paragraphs, we turn to the study of random and generic oracles for help in hypothesis formation.<sup>8</sup>

#### 4.1 Random and Generic Oracles

The study of random oracles in complexity theory was initiated in [BG81]. There, among other results, it was shown that with probability one,  $P^A \neq NP^A$ , relative to a random oracle  $A$ . More formally, if we consider a probability space of all oracles over the alphabet  $\Sigma = \{0, 1\}$ , where for each string  $x$  the event  $x \in A$  has probability one half and these probabilities are independent of each other, then the set of all oracles  $A$  relative to which  $P \neq NP$  has measure one. Bennett and Gill observed in [BG81] that for most complexity-theoretic statements  $\mathcal{C}$  of interest (e.g., for  $\mathcal{C}$  equal to any of the statements “ $P=BPP$ ,” “ $P=NP \cap coNP$ ,”

<sup>8</sup> Some of the material in Section 4.1 originally appeared in [All89b].

etc.), the set of oracles relative to which  $\mathcal{C}$  holds satisfies Kolmogorov’s zero-one law (see, e.g., [Oxt80](Theorem 21.3)). As a consequence, for statements  $\mathcal{C}$  of this sort, either  $\mathcal{C}$  holds with probability one or with probability zero, relative to a random oracle.

Bennett and Gill went on to conjecture that complexity theoretic statements that hold with probability one relative to a random oracle also hold in the unrelativized case. Although their conjecture has been disproved [Kur83, HCRR90, CGH90], it is at least true that the study of complexity-theoretic statements that hold relative to a random oracle provides an internally consistent world view, which sometimes seems useful in gaining intuition concerning the unrelativized case.<sup>9</sup> Thus we will examine the  $K_L$  complexity of sets  $L$  in  $P$ , relative to random oracles.

An alternative to random oracles is provided by the notion of “generic” oracles. Generic sets arise in set theory, logic, and computability, as examples of sets that simultaneously satisfy all properties that can be guaranteed via certain types of diagonalization arguments. They also arise from the use of Baire category to describe a topological notion of “typical set,” analogous to the measure-theoretic notion of randomness. The following paragraphs provide a brief introduction to genericity; the reader is encouraged to consult the cited references for more detailed discussions.

The fundamental notion of Baire category is the concept of a *nowhere-dense* set; a set  $\mathcal{C}$  (on the real line, say) is nowhere-dense if it is “full of holes” in the sense that for every interval  $I$  there is a subinterval  $J$  contained in the complement of  $\mathcal{C}$ . (The classic example of a nowhere-dense set is the Cantor “excluded middle” set; nowhere dense sets should intuitively be thought of as being “small” in some sense.) For our purposes, the nowhere-dense sets  $\mathcal{C}$  we will be considering will be sets of oracles, and intervals correspond to sets of oracles, all of which agree on some finite initial segment. Thus  $\mathcal{C}$  is nowhere-dense if for every finite oracle  $F$ , there is a finite extension  $F' \supseteq F$  such that every extension of  $F'$  is in the complement of  $\mathcal{C}$ . This corresponds to the classical definition (e.g., as presented in [Oxt80]), but for our purposes we will want to require that the function (called an *extension function*) that produces  $F'$  from input  $F$  be describable in first-order logic, or even be computable. A development along this line leads to notions of effective and resource-bounded category (see [Meh73, Lut90, Fen91]). Let  $\Delta$  be a class of extension functions. Then an oracle  $G$  is  $\Delta$ -generic if  $G$  is not an element of any  $\Delta$ -nowhere-dense set. Equivalently,  $G$  is  $\Delta$ -generic if for all extension functions  $h \in \Delta$ , there is some finite oracle  $F$  such that  $h(F)$  is an initial segment of  $G$ .

Many diagonalization arguments can be modelled in terms of extension functions. For example, a typical diagonalization argument proceeds in stages, with  $F_0 = \emptyset$  at stage 0, and then, given any finite oracle  $F_i$  at the start of stage  $i$ , the argument shows how to build an extension  $F_{i+1}$  satisfying some property. That

<sup>9</sup> A number of other papers discuss in depth the interpretation that should be given to results concerning random oracles. The reader is referred to [KMR89, KMR91, Cai89, Boo91].

is, the construction is the description of an extension function. Furthermore, if  $\Delta$  is a class of extension functions (such as the class of extension functions describable in first-order logic, the class of recursive extension functions, etc.) and if  $G$  is  $\Delta$ -generic, then  $G$  satisfies all properties that can be ensured via diagonalization arguments in  $\Delta$ . For example, if  $\Delta$  is the class of recursive extension functions and  $G$  is  $\Delta$ -generic, then  $P^G \neq NP^G \neq coNP^G$  (because the standard diagonalization argument showing the existence of oracles satisfying this property [BGS75] can be modelled in this way). Different notions of genericity (for different classes  $\Delta$ ) have been studied by [Maa82, AFH87, Dow82, Poi86, BI87, Fen91]. Observe that if  $\Delta \subseteq \Delta'$ , then  $G$   $\Delta'$ -generic implies  $G$   $\Delta$ -generic.

In [BI87] Blum and Impagliazzo promoted the study of complexity classes relative to generic oracles specifically as an alternative to random oracles. (They focused primarily on the notion of genericity that results when  $\Delta$  is the class of extension functions expressible in the first-order theory of arithmetic; for the rest of this paper, “generic” will mean  $\Delta$ -generic for this choice of  $\Delta$ .)

As with random oracles, generic oracles offer a consistent world view, in that all “reasonable” complexity theoretic statements  $\mathcal{C}$  either hold relative to all generic oracles or hold relative to no generic oracle. In [BI87], Blum and Impagliazzo make the case that generic oracles are perhaps more likely than random oracles to give correct intuition concerning inclusions among complexity classes; they prove a number of theorems to support this case. They also show that it will be impossible to determine if certain questions (such as  $P = NP \cap coNP$ ) hold relative to a generic oracle, without first solving some related questions in the *unrelativized* case. In fact, at the time of this writing, there is no statement  $\mathcal{C}$  concerning inclusions among “familiar” complexity classes that is known to hold relative to a generic oracle and known not to hold relative to a random oracle, or vice-versa. Furthermore, the statements  $\mathcal{C}$  that are shown in [HCRR90, CGH90] to hold relative to a random oracle but to be false in the unrelativized case, also hold relative to generic oracles. That is, neither random nor generic oracles give reliable information about which statements hold in the unrelativized case.

The reader is certainly asking “Then why consider random and generic oracles at all?”

Our purpose here is to investigate the question of whether or not there is any relationship between the density of a set  $L$  in  $P$  and the upper bounds that one can prove on  $K_L$ . We have seen above that certain popular conjectures indicate that such a relationship does exist. We shall see below that relative to a random oracle there is in fact a *very* close relationship between the density of a set  $L$  in  $P$  and the growth rate of  $K_L$ . On the other hand, relative to a generic oracle there is *no* such relationship at all. We leave the interpretation of these results to the reader; let us just mention here, however, that we conjecture that one of these extremes actually holds in the unrelativized case. That is, we believe that either there is a *close* connection between density and Kolmogorov complexity of sets, or there is *no* connection at all.

**Theorem 11.** *For a large class of functions  $f$ , relative to a random oracle  $A$ :*

- (a)  $(L \in P^A/\text{poly and } d_L(n) \geq 1/f(n)) \Rightarrow K_L^A(n) \leq \log f(n) + O(\log n)$ .  
 (b)  $\exists L \in P^A, d_L(n) \geq 1/f(n) \text{ and } K_L^A(n) \geq (\log f(n))/5 \Leftrightarrow 2 \log n$ .

(That is, relative to a random oracle, sets  $L$  of density  $1/f(n)$  can have  $K_L(n)$  no greater than about  $\log f(n)$ , and the bound is relatively tight. In the statement of this theorem,  $K_L^A$  is simply the function that one obtains from the definition of  $K_L$ , where the universal machine has access to the oracle  $A$ .)

*Proof.* In order to prove part 1, it suffices to show that, for all  $\delta > 0$ , the set of oracles  $\mathcal{C}$  has measure less than  $\delta$ , where

$$\mathcal{C} = \{A : \exists L \in P^A/\text{poly } d_L(n) \geq 1/f(n) \text{ and } \forall c \exists^\infty n K_L^A(n) > \log f(n) + c \log n\}.$$

The idea of the proof is to show that if a machine accepts very many strings relative to a random oracle, then we can find some accepted string “encoded” in the oracle, in the following sense. Given the numbers  $n$  and  $r$ , one can query an oracle about membership for the strings

$$y_{2^{n+rn+1}}, y_{2^{n+rn+2}}, \dots, y_{2^{n+rn+n}},$$

where  $y_1, y_2, \dots$  is a lexicographic enumeration of  $\Sigma^*$ . These  $n$  queries return  $n$  answers from the oracle, and these answers can be concatenated to form a string  $w$  that can be said to be “encoded” in the oracle, with index  $\langle n, r \rangle$ . Thus  $w$  has relatively low Kt-complexity, relative to the oracle. (The actual encoding used will vary only slightly from this.)

Let  $\delta$  be given, and let  $D = \log(1/\delta)$ . In the following discussion, assume that  $M_1, M_2, \dots$  is an enumeration of polynomial-time oracle Turing machines, where  $M_i$  runs in time  $n^i + i$ .

Define  $E_{i,n} = \{A : \text{there is an “advice” string } z \text{ of length } n^i \text{ with which } M_i^A \text{ accepts } \geq 2^n/f(n) \text{ strings of length } n \text{ and does not accept any of the } f(n)(i+n+D) \text{ strings of length } n \text{ given by the characteristic sequence of } A \text{ starting at } 0^{n^i+i+1}\}$ . That is,  $A \in E_{i,n}$  if the oracle machine  $M_i$ , given some advice string  $z$  for the strings of length  $n$ , accepts many strings of length  $n$ , but nonetheless manages to avoid accepting any of the strings that are stored in the “table” encoded by the oracle at position  $0^{n^i+i+1}$ . Since the strings encoded in this “table” can’t actually be read by  $M_i$  on inputs of length  $n$  (because it doesn’t have time to query the oracle about strings of that size), acceptance of each one of those strings occurs with probability at least  $1/f(n)$  (since  $M_i$  is accepting at least a fraction of  $1/f(n)$  of the strings of length  $n$ ). It follows that each  $E_{i,n}$  has measure at most  $(1 \Leftrightarrow \frac{1}{f(n)})^{f(n)(i+n+D)} < 2^{-(i+n+D)}$ . Thus  $\bigcup_{i,n} E_{i,n}$  has measure less than  $< \delta$ .

Note that each of the  $f(n)(i+n+\log \delta)$  strings of length  $n$  encoded in  $A$  starting at  $0^{n^i+i+1}$  can be described relative to  $A$  by the pair  $\langle n, j \rangle$  for some  $j \leq f(n)(i+n+\log \delta)$ , and thus any such string has  $\text{Kt}^A$  complexity bounded by  $O(\log n) + \log f(n) + O(1)$ . Thus  $A \in \mathcal{C}$  implies there is some  $i$  such that

for infinitely many  $n$  there is an “advice” string  $z$  of length  $n^i$  on which  $M_i^A$  accepts at least  $2^n/f(n)$  strings of length  $n$  but does not accept any of the strings appearing soon after  $0^{n^i+i+1}$  in the encoding given by  $A$ . Thus  $\mathcal{C} \subseteq \bigcup_{i,n} E_{i,n}$ . The result follows.

To see part 2, note that, with probability one, there is a set  $B$  in  $\text{NTIME}^A(2^n)$  that has no infinite subset in  $\text{DTIME}^A(2^{2^{n-1}})$  [BG81, Gas87]. It follows that the corresponding NE predicate is immune with respect to  $\text{DTIME}^A(2^{2^{n-1}})$ , and thus, as we observed earlier, with probability one there is a set  $C$  in  $P^A$  with  $K_C^A(n) \geq n/5$  for all large  $n$  such that  $K_C^A(n)$  is defined.

Now let  $f$  be any easy-to-compute function of the form  $f(n) = 2^{g(n)}$ , and let  $L = \{y : y = xz \text{ for some } z \in C \text{ with } |z| = g(|y|)\}$ . Then  $d_L(n) \geq 1/f(n)$ , and  $K_L(n) \geq g(n)/5 \Leftrightarrow 2 \log n$ .  $\square$

In [All89a], some hope is held out that it might be possible to show that there are relatively dense sets  $L$  in  $P/\text{poly}$  such that  $K_L(n)$  grows somewhat quickly. (There would have been interesting consequences for the theory of pseudorandom generators, if such sets could have been shown to exist.) The preceding theorem dashes these hopes (at least as far as relativizing proof techniques are concerned), since the sort of bounds that [All89a] discussed the possibility of exceeding are exactly the bounds shown above to hold relative to a random oracle.

Theorem 11 shows that, relative to a random oracle, there is a very close relationship between the density of a set and the achievable Kt-complexity of the simplest elements of the set. Next we shall see that, relative to a generic oracle, no such relationship exists.

**Theorem 12.** *Relative to a generic oracle  $A$ ,*

- (a) *There is set  $L$  in  $P^A$  such that, for infinitely many  $n$ ,  $d_L(n) \geq 1 \Leftrightarrow 2^{-n/2}$  and  $K_L^A(n) \geq n/4$ .*
- (b) *For all infinite  $L$  in  $P^A$ ,  $K_L^A(n) \neq \omega(\log n)$ .*

Thus, relative to a generic oracle, there are sets  $L$  that, for infinitely many  $n$ , contain many strings of length  $n$ , but only contain complex strings of length  $n$ . However, every infinite set in  $P^A$  contains infinitely many simple strings.

*Proof.* Part 2 follows from a proof very similar to that of Theorem 2.7 in [BI87].

To see part 1, we let  $L$  be the generic oracle  $A$  itself. Thus, we need to show that, for a generic oracle  $A$ , there are infinitely many  $n$  such that  $d_A(n) \geq 1 \Leftrightarrow 2^{-n/2}$ , and  $K_A^A(n) \geq n/4$ .

Let  $G$  be any finite oracle. Let  $n$  be chosen so that  $G$  has no string of length  $n$  or greater. For all strings  $w$  of length  $\leq n/4$ , run  $M_u^G(w)$  for  $2^{n/4}$  steps, and let  $Q$  be the set of strings output or queried by  $M_u$  during any of these computations. Note that  $\|Q\| \leq 2^{n/2}$ . Let  $G' = G \cup \{x \in \Sigma^n : x \text{ is not in } Q\}$ . By construction,  $G'$  contains many strings of length  $n$ , but contains no string of length  $n$  with Kt<sup>A</sup>-complexity  $\leq n/4$ . It follows by the results and definitions of [BI87] that, since any finite oracle can be extended in this way, any generic oracle has the properties claimed in the statement of the theorem.  $\square$

**Corollary 13.** *Relative to a generic oracle, there is no pseudorandom generator that is secure against P/poly adversaries.*

*Proof.* It follows easily from the preceding theorem that relative to a generic oracle  $A$  there is a dense set  $L$  in  $P^A/\text{poly}$  such that  $K_L^A(n) \geq n/4$  for infinitely many  $n$ . The result now follows from Theorem 10.  $\square$

Although we turned to generic and random oracles in order to find help in guiding our intuition concerning the likely behavior of the functions  $K_L$  for sets  $L$  in  $P$  and  $P/\text{poly}$ , the results of this section have given contrary indications. It is still far from clear how one might expect the  $K_L$  complexity of sets  $L$  in  $P$  to behave.

## 5 Circuit Complexity

Recently, Kolmogorov complexity has been used as a tool in proving some new results in the area of circuit complexity; we will review these developments in this section. First let us present some basic definitions. (For more background on circuit complexity the reader is referred to the excellent exposition in [BS90].)

A language  $L$  is said to be accepted by a family of circuits  $\{C_n\}$  if each circuit  $C_n$  takes inputs of length  $n$ , and for each  $x$  of length  $n$ ,  $x \in L$  iff  $C_n$  outputs 1 when given input  $x$ . The size of a circuit is the number of gates, and the depth of a circuit is the length of the longest path from input to output.

The class  $AC^0$  is the class of languages that can be recognized by families of circuits of polynomial size and constant depth, where these circuits consist of unbounded fan-in AND and OR gates. Powerful combinatorial lower bound techniques have been developed in [Hås86] (among others), showing that many very simple sets (notably the set PARITY consisting of all strings with an odd number of 1s) cannot be computed by constant depth circuits of such gates of less than exponential size.

Although  $AC^0$  is a very “weak” complexity class in some sense, note that using the definition given above,  $AC^0$  contains nonrecursive sets. (For example, there are nonrecursive tally sets, and every tally set is trivially in  $AC^0$ .) These pathological examples can be avoided by restricting our attention to “uniform” circuit families: i.e., families  $\{C_n\}$  such that the function  $n \mapsto C_n$  is “easily computable” in some sense. The issue of choosing the correct notion of uniformity for  $AC^0$  has been addressed by [BIS90]. Throughout the rest of this paper, we will consider only “uniform”  $AC^0$ .

In light of the impressive lower bound results that have been proved, showing that various languages are not in  $AC^0$ , it is natural to wonder if stronger separations of  $P$  and  $AC^0$  can be proved. For instance, is there a set in  $P$  that has no infinite subset in  $AC^0$ ? (Such a set is said to be *immune* to  $AC^0$ .) In most cases in complexity theory, if a complexity class  $\mathcal{C}_1$  can be shown to properly contain a complexity class  $\mathcal{C}_2$ , then  $\mathcal{C}_1$  can usually be shown to contain a set that is immune to  $\mathcal{C}_2$ .

Somewhat surprisingly, it seems that it will represent a significant breakthrough if one is able to present a language in P (or even in NP) that is immune to  $AC^0$  (or to show that no such language exists). As we show below, these questions are very closely related to questions about the complexity of sets in  $\mathbf{E}$ .

Note that if  $L$  is a P-printable set, then  $L$  is accepted by a family of constant-depth circuits of unbounded fan-in AND and OR gates  $\{C_n\}$  where the function  $n \mapsto C_n$  is computable in polynomial time. Call the set of languages accepted by circuits of this type P-Uniform  $AC^0$ . It is not known if P-Uniform  $AC^0 = AC^0$ ; the P-uniformity condition just described is much less restrictive than the uniformity condition of [BIS90]. This is addressed by the following theorem:

**Theorem 14.** [AG91] *P-Uniform  $AC^0 = AC^0$  iff  $\mathbf{E} = \bigcup_k \Sigma_k \text{time}(n)$ .*<sup>10</sup>

This may be taken as evidence that  $AC^0$  is properly contained in P-uniform  $AC^0$ , because  $\mathbf{E} = \bigcup_k \Sigma_k \text{time}(n)$  implies that  $\mathbf{E} = \text{DSPACE}(n) = \text{NSPACE}(n)$ , as well as implying that the polynomial hierarchy collapses and is equal to PSPACE (which in turn is equal to  $\text{DTIME}(2^{n^{O(1)}})$ ).

Recall that every NE predicate is solvable in exponential time if and only if every set in NP has an infinite P-printable subset. By the observations in the preceding paragraph, this happens only if every set in NP has an infinite subset in P-Uniform  $AC^0$ . Thus we observe:

**Observation 15.** *If every NE predicate is solvable in exponential time and  $\mathbf{E} = \bigcup_k \Sigma_k \text{time}(n)$ , then no set in NP is immune to  $AC^0$ .*

The hypothesis of this observation seems quite unlikely;  $\bigcup_k \Sigma_k \text{time}(n)$  appears to be a small subclass of  $\mathbf{E}$ , and they can be equal only if the polynomial hierarchy collapses. Assuming  $\mathbf{E} = \bigcup_k \Sigma_k \text{time}(n)$  thus says that  $\mathbf{E}$  is “not very powerful” in some sense, and it seems difficult to imagine that exponential time could simultaneously be powerful enough to solve all NE predicates. Nonetheless, it is shown in [AG91] that:

**Theorem 16.** [AG91] *There is an oracle relative to which all NE predicates are solvable in exponential time and  $\mathbf{E} = \bigcup_k \Sigma_k \text{time}(n)$ .*

Thus it would represent a significant advance at this time to show that there are sets in NP that are immune to  $AC^0$ . The oracle construction in [AG91] makes heavy use of the notions of time-bounded Kolmogorov complexity surveyed here.

Conversely, it was observed in [AG91] that if  $\text{P}=\text{NP}$ , then there is a set in P that is immune to  $AC^0$  (because  $\text{P}=\text{NP}$  implies there is a set in  $\mathbf{E}$  that is immune to  $\bigcup_k \Sigma_k \text{time}(n)$ , and this gives rise to a tally set in P that is immune to  $AC^0$ ). Thus it will also be very significant if one can show that there are *no* sets in NP that are immune to  $AC^0$ .

<sup>10</sup>  $\Sigma_k \text{time}(n)$  is the linear-time analog of the  $k^{\text{th}}$  level of the polynomial-time hierarchy.

## 6 Conclusion

We have studied one method of measuring the time-bounded Kolmogorov complexity of sets, and we have surveyed a number of applications of this measure to different topics in complexity theory. We believe that functions of the form  $K_L$  offer a useful way of visualizing the complexity of a set, and we hope that they will prove useful in more situations to come.

## 7 Acknowledgments

I thank Ken Regan and Jack Lutz for helpful comments regarding genericity.

## References

- [All85] E. Allender. *Invertible Functions*. PhD thesis, Georgia Institute of Technology, 1985.
- [All89a] E. Allender. Some consequences of the existence of pseudorandom generators. *J. Comput. System Sci.* 39:101–124, 1989.
- [All89b] E. Allender. The generalized Kolmogorov complexity of sets. In *Proc. 4th IEEE Structure in Complexity Theory Conf.*, pages 186–194, 1989.
- [AG91] E. Allender and V. Gore. On strong separations from  $AC^0$ . In *Proc. Fundamentals of Computation Theory*, Springer-Verlag, Lecture Notes in Computer Science 529:1–15, 1991.
- [AR88] E. Allender and R. Rubinfeld. P-printable sets. *SIAM J. Comput.* 17:1193–1202, 1988.
- [AW90] E. Allender and O. Watanabe. Kolmogorov complexity and degrees of tally sets. *Inform. and Computation* 86:160–178, 1990.
- [AFH87] K. Ambos-Spies, H. Fleischhack, and H. Huwig. Diagonalizations over polynomial-time computable sets. *Theoret. Comput. Sci.* 51:177–204, 1987.
- [BGS75] T. Baker, J. Gill, and R. Solovay. Relativizations of the  $P=?NP$  question. *SIAM J. Comput.* 4:431–444, 1975.
- [BDG87] J. Balcázar, J. Díaz, and J. Gabarró. Characterizations of the class PSPACE/poly. *Theoret. Comput. Sci.* 52:251–267, 1987.
- [BG81] C. Bennett and J. Gill. Relative to a random oracle,  $P(A) \neq NP(A) \neq Co-NP(A)$  with probability 1. *SIAM J. Comput.* 10:96–113, 1981.
- [BGM90] A. Bertoni, M. Goldwurm, and P. Massazza. Counting problems and algebraic formal power series in noncommuting variables. *Inform. Processing Letters* 34:117–121, 1990.
- [BGS91] A. Bertoni, M. Goldwurm, and N. Sabadini. The complexity of computing the number of strings of given length in context-free languages. *Theoret. Comput. Sci.* 86:325–342, 1991.
- [BIS90] D. Mix Barrington, N. Immerman, and H. Straubing. On uniformity within  $NC^1$ . *J. Comput. System Sci.* 41:274–306, 1990.
- [Blu67] M. Blum. A machine-independent theory of the complexity of recursive functions. *J. ACM* 14:322–336, 1967.
- [BI87] M. Blum and R. Impagliazzo. Generic oracles and oracle classes. In *Proc. 28th IEEE Symp. Foundations of Computer Science*, pages 118–126, 1987.

- [Boo91] R. Book. Some observations on separating complexity classes. *SIAM J. Comput.* 20:246–258, 1991.
- [Boo92] R. Book. On sets with small information content. In this volume.
- [BH88] R. Boppana and R. Hirschfeld. Pseudorandom generators and complexity classes. In *Advances in Computing Research. Volume 5: Randomness and Computation*, pages 1–26. Edited by S. Micali. JAI Press, Greenwich, CT, 1988.
- [BS90] R. Boppana and M. Sipser. The complexity of finite functions. In *Handbook of Theoretical Computer Science. Vol. A: Algorithms and Complexity*, pages 757–804. Edited by J. van Leeuwen. MIT Press (in the United States, Canada, and Japan), and Elsevier Science Publishers (in other countries), 1990.
- [Cai89] J.-Y. Cai. With probability one, a random oracle separates PSPACE from the polynomial-time hierarchy. *J. Comput. System Sci.* 38:68–85, 1989.
- [CGH90] B. Chor, O. Goldreich, and J. Håstad. The random oracle hypothesis is false. Technical report 631, Department of Computer Science, Technion – Israel Institute of Technology, 1990.
- [Dow82] M. Dowd. Forcing and the P hierarchy. Technical report LCSR-TR-35, Laboratory for Computer Science Research, Rutgers University, 1982.
- [Fen91] S. Fenner. Notions of resource-bounded category and genericity. In *Proc. 6th IEEE Structure in Complexity Theory Conf.*, pages 196–212, 1991.
- [Gas87] W. Gasarch. Oracles: three new results. In *Mathematical Logic and Theoretical Computer Science*, Marcel Dekker, Inc., Lecture Notes in Pure and Applied Mathematics 106:219–251, 1987.
- [GS91] A. Goldberg and M. Sipser. Compression and ranking. *SIAM J. Comput.* 20:524–536, 1991.
- [Gor90] V. Gore. Personal communication.
- [Har83] J. Hartmanis. Generalized Kolmogorov complexity and the structure of feasible computations. In *Proc. 24th IEEE Symp. Foundations of Computer Science*, pages 439–445, 1983.
- [HCR90] J. Hartmanis, R. Chang, D. Ranjan, and P. Rohatgi. Structural complexity theory: recent surprises. In *Proc. 2nd Scandinavian Workshop on Algorithm Theory*, Springer-Verlag, Lecture Notes in Computer Science 447:1–12, 1990.
- [HH88] J. Hartmanis and L. Hemachandra. On sparse oracles separating feasible complexity classes. *Inform. Processing Letters* 28:291–296, 1988.
- [HY84] J. Hartmanis and Y. Yesha. Computation times of NP sets of different densities. *Theoret. Comput. Sci.* 34:17–32, 1984.
- [Hås86] J. Håstad. *Computational Limitations for Small-Depth Circuits*. MIT Press, 1986.
- [Hås90] J. Håstad. Pseudo-random generators under uniform assumptions. In *Proc. 22nd ACM Symp. Theory of Computing*, pages 395–404, 1990.
- [HR90] L. Hemachandra and S. Rudich. On the complexity of ranking. *J. Comput. System Sci.* 41:251–271, 1990.
- [Huy85] D. Huynh. Non-uniform complexity and the randomness of certain complete languages. Technical report TR 85-34, Computer Science Department, Iowa State University, 1985.
- [Huy86] D. Huynh. Resource-bounded Kolmogorov complexity of hard languages. In *Proc. Structure in Complexity Theory*, Springer-Verlag, Lecture Notes in Computer Science 223:184–195, 1986.

- [Huy90a] D. Huynh. The complexity of ranking simple languages. *Math. Systems Theory* 23:1–20, 1990.
- [Huy90b] D. Huynh. Efficient detectors and constructors for simple languages. Technical report UTDCS-26-90, Computer Science Program, University of Texas at Dallas, 1990.
- [IT89] R. Impagliazzo and G. Tardos. Decision versus search in super-polynomial time. In *Proc. 30th IEEE Symp. Foundations of Computer Science*, pages 222–227, 1989.
- [ILL89] R. Impagliazzo, L. Levin, and M. Luby. Pseudo-random generation from one-way functions. In *Proc. 21st ACM Symp. Theory of Computing*, pages 12–24, 1989.
- [KGY89] M. Kharitonov, A. Goldberg, and M. Yung. Lower bounds for pseudorandom number generators. In *Proc. 30th IEEE Symp. Foundations of Computer Science*, pages 242–247, 1989.
- [Ko86] K. Ko. On the notion of infinite pseudorandom sequences. *Theoret. Comput. Sci.* 48:9–33, 1986.
- [Kur83] S. Kurtz. Notions of weak genericity. *J. Symbolic Logic* 48:764–770, 1983.
- [KMR89] S. Kurtz, S. Mahaney, and J. Royer. The isomorphism conjecture fails relative to a random oracle. In *Proc. 21st ACM Symp. Theory of Computing*, pages 157–166, 1989.
- [KMR91] S. Kurtz, S. Mahaney, and J. Royer. Average dependence and random oracles. Technical report SU-CIS-91-03, School of Computer and Information Science, Syracuse University, 1991.
- [Kur83] S. Kurtz. On the random oracle hypothesis. *Inform. and Control* 57:40–47, 1983.
- [Lev73] L. Levin. Universal sequential search problems. *Problems Inform. Transmission* 9:265–266, 1973.
- [Lev84] L. Levin. Randomness conservation inequalities; information and independence in mathematical theories. *Inform. and Control* 61:15–37, 1984.
- [LV90] M. Li and P. Vitányi. Applications of Kolmogorov Complexity in the Theory of Computation. In *Complexity Theory Retrospective*, pages 147–203. Edited by A. Selman. Springer-Verlag, 1990.
- [Lut90] J. Lutz. Category and measure in complexity classes. *SIAM J. Comput.* 19:1100–1131, 1990.
- [Lut91] J. Lutz. Almost everywhere high nonuniform complexity. *J. Comput. System Sci.*, to appear. A preliminary version appeared in *Proc. 4th IEEE Structure in Complexity Theory Conf.*, pages 37–53, 1989.
- [Maa82] W. Maass. Recursively enumerable generic sets. *J. Symbolic Logic* 47:809–823, 1982.
- [Meh73] K. Mehlhorn. On the size of sets of computable functions. In *Proc. 14th IEEE Symp. Switching and Automata Theory*, pages 190–196, 1973.
- [MS90] M. Mundhenk and R. Schuler. Non-uniform complexity classes and random languages. In *Proc. 5th IEEE Structure in Complexity Theory Conf.*, pages 110–119, 1990.
- [Oxt80] J. Oxtoby. *Measure and Category* (second edition). Springer-Verlag, 1980.
- [Poi86] B. Poizat.  $Q = NQ$ ? *J. Symbolic Logic* 51:22–32, 1986.
- [SF90] L. Sanchis and M. Fulk. On the efficient generation of language instances. *SIAM J. Comput.* 19:281–296, 1990.
- [Sip83] M. Sipser. A complexity theoretic approach to randomness. In *Proc. 15th ACM Symp. Theory of Computing*, pages 330–335, 1983.

- [Yao82] A. Yao. Theory and applications of trapdoor functions. In *Proc. 23rd IEEE Symp. Foundations of Computer Science*, pages 80–91, 1982.