Electronic Colloquium on Computational Complexity, Report No. 144 (2005)

Time-Bounded Universal Distributions

Luis Antunes* *University of Porto*

Lance Fortnow† *University of Chicago*

5th December 2005

Abstract

We show that under a reasonable hardness assumptions, the time-bounded Kolmogorov distribution is a universal samplable distribution. Under the same assumption we exactly characterize the worst-case running time of languages that are in average polynomial-time over all P-samplable distributions.

1 Introduction

Leonid Levin [Lev73] considered a semi-measure defined by $\mu(x) = 2^{-K(x)}$ where $K(x)$ is the prefix-free Kolmogorov complexity of x. Levin showed that $\mu(x)$ is universal among the semi-computable semi-measures.

In this paper we consider the time-bounded variation $\mu^t(x) = 2^{-K^t(x)}$. This measure seems to sit somewhere between the polynomial-time computable and polynomial-time samplable distributions. We show that under reasonable hardness assumptions $\mu^t(x)$ is universal among the polynomial-time samplable distributions, specifically

1. For any polynomial p, there is a polynomial-time samplable distribution τ and a constant c such that $\tau(x) \geq c\mu^p(x)$ for every x.
2. If $\mathbf{E} = \mathbf{DTIME}(2^{O(n)})$ does not have size $2^{o(n)}$ circuits with Σ_2^p gates then for every polynomial-time samplable distribution τ there is a polynomial p such that $\mu^p(x) \geq p(|x|)\tau(x)$.

The first item was proved earlier by Antunes, Fortnow and Vinodchandran [AFV03]. Our paper focuses on proving the second item.

Building on Antunes, Fortnow and Vinodchandra [AFV03] we can characterize the worst-case running time of languages that run in average polynomial-time over the samplable distributions. We show that if $\mathbf{E} = \mathbf{DTIME}(2^{O(n)})$ does not have size $2^{o(n)}$ circuits with Σ_2^p gates then the following are equivalent for all languages L.

*This author is also partially supported by funds granted to LIACC through the Programa de Financiamento Plurianual, Fundação para a Ciência e Tecnologia and Programa POSI. Email: `lfa@ncc.up.pt`. Web: `http://www.ncc.up.pt/~lfa`.

†Email: `fortnow@cs.uchicago.edu`. Web: `http://people.cs.uchicago.edu/~fortnow`.

ISSN 1433-8092

1. L is polynomial-time on σ-average for all polynomial-time samplable σ.

2. For all polynomials p, there is an algorithm computing the language L using time $t(x) = 2^{O(K^p(x)-K(x)+\log|x|)}$.

Our proof techniques use recent advances in pseudorandom generators based on hard functions, specifically the work of Nisan and Wigderson [NW94], Impagliazzo and Wigderson [IW97] and Klivans and van Melkebeek [KvM99].

2 Preliminaries

We use binary alphabet $\Sigma = \{0,1\}$ for encoding strings. Our computation model will be *prefix free* Turing machines: Turing machines with a one-way input tape (the input head can only read from left to right), a one-way output tape and a two-way work tape. The function log denote $\log_2$. All explicit resource bounds we use in this paper are time-constructible.

2.1 Kolmogorov Complexity

We give essential definitions and basic result in Kolmogorov complexity and refer the reader to the textbook by Li and Vitányi [LV97] for more details. We are interested in self-delimiting Kolmogorov complexity. A set of strings A is prefix-free if there are no strings x and y in A where x is a proper prefix of y.

Definition 2.1 *Let U be a fixed Turing machine with a prefix-free domain. Then for any string $x, y \in \{0,1\}^*$, the Kolmogorov complexity of x given y is, $K(x|y) = \min_p\{|p| : U(p,y) = x\}$.*
For any time constructible t, the t-time-bounded Kolmogorov complexity of x given y is, $K^t(x|y) = \min\{|p| : U(p,y) = x$ in at most $t(|x|)$ steps$\}$.

The default value for y is the empty string ϵ, we typically drop this argument in the notation. We can fix a universal machine U whose program size $|p|$ is at most a constant additive factor worse, and the running time t at most a logarithmic multiplicative factor.

A function $f : S \longrightarrow [0,1]$ is called a *semi-measure* over the space S if

$$\sum_x f(x) \leq 1$$

It is called a *measure* if equality holds. A semi-measure is called *constructive* if it is semi-computable from below, i.e., there is a computable function $g(x,t)$ monotone in t such that $f(x) = \lim_{t\to\infty} g(x,t)$.

A constructive semi-measure which multiplicatively dominates every other constructive semi-measure is called *universal*. In the unbounded case, an enumeration $\mu_1, \mu_2, \cdots$ of r.e., semi-measures can be used to define a universal semi-measure $\mathbf{m}$. For all $x \in \Sigma^*$ let

$$\mathbf{m}(x) = \sum_n \alpha(n) \times \mu_n(x)$$

where α is some function such that $\sum_n \alpha(n) \leq 1$.

We can use Kolmogorov complexity to define a semi-measure.

Definition 2.2 *The algorithmic probability $R(x)$ of a string x is defined as*

$$R(x) = 2^{-K(x)}.$$

$R(x)$ is a semi-measure because of Kraft's inequality that states that $\sum_{y \in A} 2^{-|y|} \leq 1$ for any prefix-free set A.

The a priori probability, based on the set of all programs producing a string x, can be thought as the probability that the universal Turing machine computes the output x if its input is provided by successive tosses of a fair coin.

Definition 2.3 *The a priori probability $Q(x)$ of a string x is defined as*

$$Q(x) = \sum_{p} 2^{-|p|}$$

where p ranges over all strings such that $U(p) = x$ but $U(p')$ does not stop for any prefix p' of p.

Levin [Lev74] and Gács [Gac74], showed that the universal distribution coincides up to a multiplicative factor with the algorithmic probability or the a priori probability. So, if x has high probability because it has many long descriptions then it must have a short description too.

Theorem 2.4 (Coding Theorem) *There is a constant c such that for every x,*

$$-\log \mathbf{m}(x) = -\log Q(x) = -\log R(x)$$

with equality up to an additive constant c

The question whether a similar result holds in the time bound setting is open. It is not known if there exist strings having no small fast programs, even though they have enough large fast programs to contribute a significant fraction of their algorithmic probability.

The definition of the t-time bounded universal distribution based on the Kolmogorov complexity is controversial, some authors define it as $R^t(x)$ and others as $Q^t(x)$. We define it as:

Definition 2.5 *The t-time bounded universal distribution, $\mathbf{m}^t$ is given by $\mathbf{m}^t(x) = 2^{-K^t(x)}$.*

2.2 P-Computable and P-Samplable distributions

Usually simple distributions are identified with the polynomial-time computable distributions.

Definition 2.6 *Let t be a time constructible function. A probability distribution function μ on $\{0,1\}^*$ is said to be t-time computable, if there is a deterministic Turing machine that on every input x and a positive integer k, runs in time $t(|x| + k)$, and outputs a fraction y such that $|\mu^*(x) - y| \leq 2^{-k}$.*

The most controversial definition in the average case complexity theory is the association of the class of *simple* distributions with P-computable, which may seem too restricting. Ben-David *et al.* in [BDCGL92] introduced a wider family of natural distributions, P-samplable, consisting of distributions that can be sampled by randomized algorithms, working in time polynomial in the length of the sample generated.

Definition 2.7 *A probability distribution μ on $\{0,1\}^*$ is said to be* P*-samplable, if there is a probabilistic Turing machine M which on input 0^k produces a string x such that $|Pr(M(0^k) = x) - \mu(x)| \leq 2^{-k}$ and M runs in time $poly(|x| + k)$.*

Every P-computable distribution is also P-samplable, however the converse is unlikely.

Theorem 2.8 ([BDCGL92]) *If one-way functions exists, then there is a* P*-samplable probability distribution μ which is not dominated by any polynomial-time computable probability distribution ν.*

One important property of $\mathbf{m}^t$ is that it dominates certain computable distributions.

Theorem 2.9 ([LV97]) *$\mathbf{m}^t$ dominates any t/n-time computable distribution.*

It is then natural to ask if there exists a polynomial-time computable distribution dominating $\mathbf{m}^t$. Schuler [Sch99] showed that if such a distribution exists then no polynomially secure pseudo-random generators exists.

Theorem 2.10 ([Sch99]) *If there exists a polynomial time computable distribution that dominates $\mathbf{m}^t$ then pseudo-random generators do not exist.*

While, it is unlikely that there are polynomial-time computable distributions dominating universal distributions, there are P-samplable distributions dominating the time-bounded universal distributions.

Theorem 2.11 ([AFV03]) *For any polynomial t, there is a* P*-samplable distribution μ which dominates $\mathbf{m}^t$.*

Proof. (Sketch) We will define a samplable distribution μ_t by prescribing a sampling algorithm for μ_t as follows. Let U be the universal machine.

Sample $n \in \mathbf{N}$ with probability $\frac{1}{n^2}$
Sample $1 \leq j \leq n$ with probability $1/n$
Sample uniformally $y \in \Sigma^j$
Run $U(y)$ for t steps. If U stops and outputs a string $x \in \Sigma^n$, output x.

For any string x of length n, $K^t(x) \leq n$. Hence it is clear that the probability that x is at least $\frac{1}{n^3} 2^{-K^t(x)}$. ◇

2.3 Pseudo-random Generators

Pseudo-random generators are efficiently computable functions which stretches a seed into a long string so that for a random input the output looks random for a resource-bounded machine. We need some pseudorandom generators based on hard functions.

Definition 2.12 *Given a Boolean function $f : \{0,1\}^* \to \{0,1\}$ and an oracle B, the circuit complexity of $C_f^B(n)$ of f at length n relative to B, is the smallest integer t such that there is a B-oracle circuit of size t that computes f on inputs of length n.*

Definition 2.13 *The hardness $H_f^B(n)$ of f at length n relative to B is the largest integer t such that, for any oracle circuit D of size at most t with n inputs*

$$|Pr_{x\in U_n}[D^B(x) = f(x)] - 1/2| < 1/t$$

Using the hardness of the parity function [Has87], Nisan and Wigderson [NW94] create a pseudo-random generator that looks random to constant depth circuits.

Lemma 2.14 (Nisan-Wigderson) *For any fixed non-negative integer d, there exists a family of generators $\{G_0, G_1, \ldots\}$ with the following properties:*

- *G_n maps strings of length u polynomial in $\log n$ to strings of length n.*
- *For any circuit D of depth d and size n, we have*

$$|\Pr_{\rho\in\{0,1\}^n}[D(\rho)] - \Pr_{\sigma\in\{0,1\}^u}[D(G_n(\sigma))]| < 1/n.$$

- *Each output bit of G_n is computable in time polynomial in $\log n$.*

Impagliazzo and Wigderson [IW97] strengthen the work of Nisan and Wigderson to show how to achieve full derandomization based on strong hardness assumptions. Klivans and van Melkebeek [KvM99] generalize Impagliazzo-Wigderson by showing the results hold for relativized worlds in a strong way.

Lemma 2.15 (Klivans-van Melkbeek) *There exists a positive constant γ and δ such that, for any oracle B, Boolean function f, and a constructible function $h : \mathbf{N} \to \mathbf{N}$ satisfying*

$$h(n) \leq (C_f^B(\gamma n))^\delta / n$$

the following holds: There exists a Boolean function g such that $H_g^B \geq h(n)$. Computing g on inputs of length n takes time $2^{O(n)}$ plus evaluating f on all inputs of length γn.

3 Main Result

In this section we show that the polynomial time bounded coding theorem holds up to a *polynomial factor*. We start showing that under a reasonable assumption the following hypothesis holds:

Hypothesis 3.1 *Consider $F : \Sigma^m \to \Sigma^n$, $n \geq m$ and $F \in$ FP. Let $T_y = \{x \in \Sigma^m : F(x) = y\}$, $r = 2^{\lfloor \log |T_y| \rfloor}$ and $V_r = \{T_z : |z| = n \text{ and } |T_z| \geq r\}$. Then there exists a pseudo-random generator G computable in time polynomial in m*

$$G : \Sigma^{m - \log(r) + O(\log(n))} \to \Sigma^m$$

such that for all $T_y \in V_r$, range(G)$\cap T_y \neq \emptyset$.

Theorem 3.2 *If there exists a language in* **DTIME**(2^n) *that does not have $2^{o(n)}$-size circuits with Σ_2-gates then Hypothesis 3.1 holds.*

The proof is done in two steps:

1. show that a random G satisfies Hypothesis 3.1.
2. derandomize. First applying Lemma 2.14 from exponential to polynomial, and then Lemma 2.15 from polynomial to logarithmic.

Proof.

Let $r = 2^{\lfloor \log |T_y| \rfloor}$, and consider a partition of Σ^m induced by the set V_r into $\frac{2^m}{r}$ pieces. By a coupon collector argument, in average, we need $\frac{2^m}{r}\log(\frac{2^m}{r})$ rounds in order to hit all of the T_z's. But

$$\begin{aligned}\frac{2^m}{r}\log(\frac{2^m}{r}) &\leq \frac{2^m}{r}\log 2^m \\ &= \frac{m2^m}{r} \\ &\leq \frac{n2^m}{r} \\ &= 2^{m+O(\log n)-\log r}\end{aligned}$$

So, with high probability a random $G : \Sigma^{m-\log(r)+O(\log(n))} \to \Sigma^m$ hit all the $\frac{2^m}{r}$ sets T_z's.

To complete the proof we show that for all y, T_y is too small or exists s such that $F(G(s)) = y$.

- if $|T_y|$ is too small ($|T_y| << 2^r$) we can approximate it.

Lemma 3.3 ([Sip83]) *Given $b, k, l > 0$, $l > \max(b, 8)$, and $C \subseteq \Sigma^k$. Randomly , select l linear functions $H = \{h_1, \cdots, h_l\}$, $h_i : \Sigma^k \to \Sigma^b$ and l^2 strings $Z = \{z_1, \cdots, z_{l^2}\} \subseteq \Sigma^b$. Then:*

1. *If $b = 2 + \lceil \log |C| \rceil$ then*
   (a) *$Pr[H(C) \geq \frac{|C|}{l}] \geq 1 - 2^l$.*
   (b) *$Pr[H(C) \cap Z \neq \emptyset] \geq 1 - 2^{l/8}$.*
2. (a) *$|H(C)| \leq l|C|$.*
   (b) *$Pr[H(C) \cap Z \neq \emptyset] \leq l^3/d$.*

Pick $2^{\frac{\epsilon r}{6}}$ independent random hash functions $f_1, \cdots, f_{2^{\frac{\epsilon r}{6}}} : \Sigma^m \to \Sigma^r$ and the $2^{\frac{\epsilon r}{3}}$ points z_i $1 \leq z_i \leq 2^{\frac{\epsilon r}{3}}$ in Σ^r. If exists an $x \in T_y$ such that $f_i(x) = z_j$ for some i, j, $1 \leq i \leq 2^{\frac{\epsilon r}{6}}$ and $1 \leq j \leq 2^{\frac{\epsilon r}{3}}$ then the set is big, otherwise it is small. Now by Lemma 3.3 we have:

- if $|T_y| \geq 2^r$ then the probability of consider it small is $\leq 2^{-\frac{2^{\frac{\epsilon r}{6}}}{8}}$.
- if $|T_y| \leq \frac{2^r}{2^{\epsilon r}} = 2^{r(1-\epsilon)}$ then the probability of consider it small is $\geq 1 - \frac{2^{\frac{\epsilon r}{2}}}{2^{\epsilon r}} = 1 - 2^{-\frac{\epsilon r}{2}}$.

- Exists s such that $F(G(s)) = y$.
  G as a random string has length $\frac{m \times n^{O(1)}}{r^{O(1)}} 2^m$, but we cannot afford $\log(\frac{m \times n^{O(1)}}{r^{O(1)}} 2^m)$ bits to describe the seed. However, we can compose the generator in Lemma 2.15, logarithmic to polynomial, with the generator in Lemma 2.14, polynomial to exponential, so we can derandomize.

  – Lemma 2.14: computed by a constant depth circuit of size $2^{2^{m^{O(1)}}}$, namely as follows. For each string y of length n, we have to verify that T_y is too small or exists an $s \in \Sigma^{m-\log r+O(\log n)}$ such that $F(G(s)) = y$. Since $F \in \mathbf{FP}$ and G is computable in time polynomial in m then $F(G())$ is computable in time polynomial in m, say m^c. Such a computation can be expressed as an OR of $2^{m-\log r+O(\log n)}$ AND´s of size m^c.

  – Lemma 2.15: recognized by circuits with Σ_2^P-oracle gates. The circuit just needs to ask the Σ_2^P oracle gate the s for a given y and verify that the s given produces a string p such that if we apply $F(p) = y$. This can be done efficiently because each bit of p can be computed from s in polynomial time and we can efficiently verify whether p produces y.

  So, under hardness assumption in Lemma 2.15 there exists a efficient pseudo-random generator rIW with logarithmic seed length that fools Σ_2^P circuits of the required polynomial size, and then using Lemma 2.14 there exists an efficient pseudo-random generator NW to compute the description of G from $rIW(s)$.

  The final description of G consists of, the seed s for rIW that is $O(\log m)$, $rIW(s)$ the seed for NW that is of size $m + O(\log n) - O(\log r)$ and some $O(\log n)$ bits to describe the parameters of rIW and NW pseudo-random generators, so at the end it requires $m + O(\log n) - O(\log k)$ bits. We can compute G as the result of $NW(rIW(s))$ and this computation takes time polynomial in n.

◇

We now state our main result

Theorem 3.4 *If there exists a language in* $\mathbf{DTIME}(2^n)$ *that does not have* $2^{o(n)}$*-size circuits with* Σ_2*-gates, then for all polynomials* t *there exists a polynomial* t' *such that for all* x*,*

$$R^{t'}(x) \geq \frac{Q^t(x)}{poly(|x|)}.$$

Proof. Consider a universal Turing machine $U : \Sigma^m \to \Sigma^n$ whose running time t is polynomial in n, i.e., U^t is in FP. For every $y \in \Sigma^n$ let

$$T_y = \{p \in \Sigma^m : U^{t(n)}(p) = y\},$$

$r = 2^{\lfloor \log |T_y| \rfloor}$ and

$$V_r = \{T_z : |z| = n \text{ and } |T_z| \geq r\}$$

By construction T_y is in V_r and by Hypothesis 3.1 there exists a pseudo-random generator computable in time polynomial in n ($t'(n)$)

$$G : \Sigma^{m-\log r+O(\log n)} \to \Sigma^m$$

such that for all $T_y \in V_r$, range$(G) \cap T_y \neq \emptyset$. Consider $\sum_{p:U^t(p)=y} 2^{-|p|}$, the length of the programs in the sum can not exceed $t(n)$, otherwise there will not be time to read the program. Fix m' that maximizes

$$Q^t_m(y) = \sum_{p:|p|=m,\ U^t(p)=y} 2^{-|p|} = \frac{|T_y|}{2^m}.$$

Note that

$$\begin{aligned} Q^t(y) &= \sum_{m=K^t(y)}^{t(n)} Q^t_m(y) \\ &\leq t(n) Q^t_{m'}(y) \end{aligned}$$

So

$$\begin{aligned} R^{t'}(y) &= 2^{-K^{t'}(y)} \\ &\geq 2^{-m' - O(\log n) + \log r} \\ &= \frac{r}{2^{m'} poly(n)} \\ &= \frac{|T_y|}{2^{m'} poly(n)} \\ &= \frac{Q^t_{m'}(y)}{poly(n)} \\ &\geq \frac{t(n) Q^t(y)}{poly(n)} \end{aligned}$$

$\diamond$

A distribution is called universal if it multiplicatively dominates every other distribution in a given class, i.e., it represents the class. Based the previous result we show that R_q is, up to a polynomial multiplicative factor, a universal P-samplable distribution.

Theorem 3.5 *If* $\mathbf{E} = \mathbf{DTIME}(2^{O(n)})$ *does not have size* $2^{o(n)}$ *circuits with* Σ^p_2 *gates then for every polynomial-time samplable distribution* σ*, exists polynomials* p *and* t *such that* $\mathbf{m}^t(x) \geq \frac{\sigma(x)}{p(|x|)}$.

Proof. Consider a P-samplable distribution σ, and a string r such that $\sigma(r) = x$ with $|r| \leq |x|$. Then

$$\mathbf{m}^t(x) = 2^{-K^t(x)} = \frac{\sum_{U^t(p)=x} 2^{-|p|}}{p(|x|)} \geq \frac{\sum_{\sigma(r)=x} 2^{-|r|}}{p(|x|)} = \frac{\sigma(x)}{p(|x|)}$$

$\diamond$

For a distribution σ, Levin [Lev86] defined a notion of polynomial-time on σ-average.

Definition 3.6 *A language* L *is polynomial-time on* σ*-average if* L *is accepted by some machine* M *whose running time is* $t(x)$ *and for some* k*,*

$$\sum_{x \in \Sigma^*} \frac{t^{1/k}(x)}{|x|} \sigma(x) < \infty$$

With abuse of notation we say a time function t is polynomial-time on σ-average if it fulfills the above equation.

Given the hardness assumption we can characterize the worst-case running time for all languages that are polynomial-time on average on P-samplable distributions.

Theorem 3.7 *If* $\mathbf{E} = \mathbf{DTIME}(2^{O(n)})$ *does not have size* $2^{o(n)}$ *circuits with* Σ_2^p *gates then,* L *is polynomial-time on* σ*-average for all* P*-samplable* σ *if and only if for all polynomials* p *the running time for some* $M(x)$ *accepting* L *is* $2^{O(K^p(x)-K(x)+log|x|)}$.

To prove Theorem 3.7 we need the following theorem due to Antunes, Fortnow and Vinodchandran [AFV03].

Theorem 3.8 (Antunes-Fortnow-Vinodchandan) *Let* T *be a constructible time bound. Then for any time constructible* t*, the following statements are equivalent.*

1. $T(x) \in 2^{O(K^t(x)-K(x)+\log n)}$.
2. T *is polynomial time on* $\mathbf{m}^t$*-average.*

We first need a lemma that if σ dominates τ and t is polynomial time on σ-average then t is polynomial-time on τ average.

Lemma 3.9 *If there is a* j *such that for all* x*,* $\sigma(x) \geq \tau(x)/|x|^j$ *and* t *is polynomial time on* σ*-average then* t *is polynomial-time on* τ *average.*

Proof. By assumption there is a k such that

$$\sum_{x\in\Sigma^*} \frac{t^{1/k}(x)}{|x|}\sigma(x) < \infty$$

Let A be the set of x such that $t(x) \geq |x|^{2jk}$. We have

$$\sum_{x\in\Sigma^*} \frac{t^{1/2jk}(x)}{|x|}\tau(x) = \sum_{x\in A} \frac{t^{1/2jk}(x)}{|x|}\tau(x) + \sum_{x\in\Sigma^*-A} \frac{t^{1/2jk}(x)}{|x|}\tau(x).$$

For the first term we have

$$\sum_{x\in A} \frac{t^{1/2jk}(x)}{|x|}\tau(x) \leq \sum_{x\in A} \frac{t^{1/2jk}(x)|x|^j}{|x|}\sigma(x) \leq \sum_{x\in A} \frac{t^{1/2jk}(x)t^{1/2k}(x)}{|x|}\sigma(x) \leq \sum_{x\in\Sigma^*} \frac{t^{1/k}(x)}{|x|}\sigma(x) < \infty.$$

For the second term we have

$$\sum_{x\in\Sigma^*-A} \frac{t^{1/2jk}(x)}{|x|}\tau(x) \leq \sum_{x\in\Sigma^*-A} \frac{|x|^{2jk/2jk}}{|x|}\tau(x) \leq \sum_{x\in\Sigma^*-A} \tau(x) \leq 1.$$

◇

We can now give the proof of Theorem 3.7.

Proof. Suppose L is polynomial-time on σ-average for all P-samplable σ. Fix a polynomial p. By Theorem 2.11 there is a P-samplable σ that dominates m^p. By Lemma 3.9, L is polynomial-time on m^p average. By Theorem 3.8 the running time of L is bounded by $2^{O(K^p(x)-K(x)+\log n)}$.

Now suppose the running time of L is bounded by $2^{O(K^p(x)-K(x)+log|x|)}$ for all polynomials p. Let σ be a P-samplable distribution. By Theorem 3.5 there are polynomials p and q such that $m^p(x) \geq \sigma(x)/q(|x|)$. By Theorem 3.8, L is polynomial time on m^p average and then by Lemma 3.9, L is polynomial time on σ-average. ◇

4 Concluding Remarks

Dieter van Melkebeek [vM05] showed how to create the pseudorandom generators we need from a slightly weaker assumption, that **E** does not have nondeterministic circuits of size $2^{o(n)}$.

Eric Allender suggested defining a version of m^t using Kolmogorov measures that build the time into the complexity instead of as a parameter. We can define $mt(x) = 2^{-Kt(x)}$ and $mT(x) = 2^{-KT(x)}$ where $Kt(x) = \min_p\{|p| + \log t \mid U(p) \text{ outputs } x \text{ in } t \text{ steps}\}$ and $KT(x) = \min_p\{|p| + t \mid U(p) \text{ outputs } x \text{ in } t \text{ steps}\}$.

Theorem 3.5 holds for mt without the need for a polynomial t, since for all functions t, $mt(x) \geq m^t(x)/t(|x|)$. However we do not know if mt is dominated by a samplable distribution.

One can build a samplable distribution that dominates mT but we do not know if Theorem 3.5 holds for that distribution.

Acknowledgments

We thank Eric Allender, Dieter van Melkebeek and Paul Vitányi for helpful discussions and comments.

References

[AFV03] L. Antunes, L. Fortnow, and V. Vinodchandran. Using depth to capture average-case complexity. In *Fundamentals of Computation Theory, 14th International Symposium, FCT*, volume 2751 of *Lecture Notes in Computer Science*, pages 303–310. Springer, 2003.

[BDCGL92] S. Ben-David, B. Chor, O. Goldreich, and M. Luby. On the theory of average case complexity. *J. Computer System Sci.*, 44(2):193–219, 1992.

[Gac74] P. Gacs. On the symmetry of algorithmic information. *Soviet Math. Dokl.*, 15:1477–1480, 1974.

[Has87] J. Hastad. *Computational limitations of small-depth circuits.* MIT Press, Cambridge, MA, USA, 1987.

[IW97] R. Impagliazzo and A. Wigderson. P=BPP unless E has subexponential circuits: derandomizing the xor lemma. In *Proceedings of the 29th STOC*, pages 220–229, 1997.

[KvM99] A. Klivans and D. van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. In *STOC '99: Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 659–667, New York, NY, USA, 1999. ACM Press.

[Lev73] L. Levin. On the notion of a random sequence. *Soviet Math. Dokl.*, 14:1413–1416, 1973.

[Lev74] L. Levin. Laws of information conservation (nongrowth) and aspects of the foundation of probability theory. *Probl. Inform. Transm.*, 10:206–210, 1974.

[Lev86] L. Levin. Average case complete problems. *SIAM J. Computing*, 15(1):285–286, February 1986.

[LV97] M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer, 1997.

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

[Sch99] R. Schuler. Universal distributions and time-bounded kolmogorov complexity. In *Proc. 16th Annual Symposium on Theoretical Aspects of Computer Science*, volume 1563, pages 434–443. Springer-Verlag, 1999.

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

[vM05] D. van Melkebeek, 2005. Personal Communication.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'