Sophistication Revisited*

Luís Antunes† Lance Fortnow‡

August 30, 2007

Abstract

Kolmogorov complexity measures the ammount of information in a string as the size of the shortest program that computes the string. The Kolmogorov structure function divides the smallest program producing a string in two parts: the *useful information* present in the string, called *sophistication* if based on total functions, and the remaining *accidental information*. We formalize a connection between *sophistication* (due to Koppel) and a variation of *computational depth* (intuitively the useful or nonrandom information in a string), prove the existence of strings with maximum sophistication and show that they are the deepest of all strings.

1 Introduction

Kolmogorov [Kol65], Solomonoff [Sol64] and Chaitin [Cha66] independently defined the complexity of a string x as the length of the shortest program that produces x. The Kolmogorov structure function divides the smallest program producing a string, as measured by the Kolmogorov complexity, in two parts: the *useful information* present in the string and the remaining *accidental information*. Kolmogorov represented the useful information by a finite set of which the object is a typical element, so that the two-stage description, the finite set together with the index of the object in that set, is as short as the shortest one-part description. Expressing the useful information as a recursive function Koppel [Kop88, KA91, Kop91] introduced the concept of *sophistication* of an object.

Bennett [Ben88] formally defined the *s-significant logical depth* of an object x as the time required by a standard universal Turing machine to generate x by a program that is no more than s bits longer than the shortest descriptions of x. Antunes *et. al.* [AFvMV06] considered logical depth as one instantiation of a more general theme, computational depth, and propose several other variants. Intuitively, *computational depth* measures the amount of "nonrandom" or "useful" information in a string. Formalizing this intuitive notion is tricky. A computationally deep string x should take a lot of effort to construct from its short description. Incompressible strings are trivially constructible from their shortest description, and therefore computationally shallow.

Related Work

Cover [Cov85] suggests that the Kolmogorov structure function is similar to "algorithmic sufficient statistics" and it is suggested that it is the algorithmic approach to the probabilistic notion of sufficient statistics. Later Gács *et. al.* [GTV01] established this relation, generalizing the Kolmogorov structure function approach to computable probability functions, the resulting theory is referred to as *Algorithmic Statistics*. Vereshchagin and Vitányi [VV02], claimed the rightness of the Kolmogorov structure function, proving that ML, MDL, and related methods in model selection always give a best possible model in complexity-restricted model classes. The most general way to proceed is perhaps to express the regularity as a recursive function. The

*Preliminary version of this paper appeared in the Proceedings of the 30th ICALP Conference [AF03]. Much of the research for this paper occurred at the NEC Research Institute.

†University of Porto. This author is partially supported by by KCrypt (POSC/EIA/60819/2004) and funds granted to LIACC through the Programa de Financiamento Plurianual, Fundação para a Ciência e Tecnologia and Programa POSI.

‡University of Chicago.

resulting measure has been called, by Koppel, the sophistication of the object. Koppel claimed that depth and sophistication for all infinite strings are equivalent. However the proof uses a different definition of depth imposing totality in the functions defining depth and defining the length of a time bound by the smallest program describing it. Vitányi [Vit06] developed the theory of recursive functions statistics, and studied its relation with the more restricted model class of finite sets, and computable probability distributions, in particular with the Kolmogorov minimal sufficient statistic.

This Work

In this paper we take a fresh look at sophistication showing that there are strings with high depth but low sophistication. This result does not contradict Koppel [Kop88] since he used a non-standard notion of logical depth. We show that there are strings with near maximum sophistication and strings of high sophistication encode the halting problem for smaller strings, i.e., they are the deepest of all strings.

We define a robust variation of sophistication, coarse sophistication, and show that it is roughly equivalent to a variation of computational depth based on the busy beaver function.

2 Preliminaries

We use binary alphabet $\Sigma = \{0,1\}$ for encoding strings. Our computation model will be Turing machines, and U denotes a fixed universal Turing machine. The function log denote $\log_2$, and we will use $|.|$ for the length of a string and also to the cardinality of a set.

Kolmogorov Complexity

We give essential definitions and basic result in Kolmogorov complexity for our needs and refer the reader to [LV97] for more details.

Definition 2.1 *A function $t : \mathbf{N} \to [0,\infty)$ is* time-constructible *if there exists a Turing machine that runs in time exactly $t(n)$ on every input of size n.*

All explicit resource bounds we use in this paper are time-constructible.

Definition 2.2 *Let U be a fixed universal Turing machine. Then for any string $x \in \{0,1\}^*$, the Kolmogorov complexity of x is, $C(x) = \min_p\{|p| : U(p) = x\}$. For any time constructible t, the t-time-bounded Kolmogorov complexity of x is, $C^t(x) = \min\{|p| : U(p) = x$ in at most $t(|x|)$ steps$\}$.*

A different universal machine U may affect the program size $|p|$ by at most a constant additive factor, and the running time t by at most a logarithmic multiplicative factor.

Definition 2.3 *A string x is* incompressible *if $C(x) \geq |x|$. We also call such x algorithmically random.*

Proposition 2.1 *For all nonnegative integers n, at least half of the strings of length at most n are incompressible.*

We extend the definition of Kolmogorov complexity to finite sets in the following way: the Kolmogorov complexity of a set S (denoted $C(S)$) is the Kolmogorov complexity of a list of its elements in some given order. As noted by Cover [Cov85], Kolmogorov proposed in 1973 at the Information Theory Symposium, Tallin, Estonia the following function:

Definition 2.4 *The Kolmogorov structure function $H_k(x|n)$ of $x \in \Sigma^n$ is defined by*

$$H_k(x|n) = \min\{\log|S| : x \in S \text{ and } C(S|n) \leq k\}.$$

Of special interest is the value

$$C^*(x|n) = \min\{k : H_k(x|n) + k = C(x|n)\}.$$

A program for x can be written in two stages:

1. `Use` p `to print the indicator function for` S.

2. `The desired string is the` ith `sequence in a lexicographic ordering of the elements of this set.`

This program has length $|p| + \log|S| + O(1)$, and $C^*(x|n)$ is the length of the shortest program p for which this two-stage description is as concise as the shortest one stage description. Note that x must be maximally random (a typical element) with respect to S otherwise the two stage description could be improved, contradicting the minimality of $C(x|n)$. Gács *et al.* [GTV01] generalize the model class from finite sets to probability distributions, where the models are computable probability density functions.

In 1982 at a seminar in the Moscow State University (see [V'y99]) Kolmogorov raised the question if "absolutely non-random" (or non-stochastic) objects exist.

Definition 2.5 *Let α and β be natural numbers. A string $x \in \Sigma^n$ is called (α, β)-stochastic if there exists a finite set S such that $x \in S$, $C(S) \leq \alpha$ and $C(x|S) \geq \log|S| - \beta$.*

The following theorem, proved by Shen [She83], is part of the answer for a corresponding problem posed by Kolmogorov about the existence of "absolutely non-random" strings.

Theorem 2.1 (Shen) *1. There exists a constant c such that, for any n and any α and β with $\alpha \geq \log n + c$ and $\alpha + \beta \geq n + 4\log n + c$, all the numbers from 0 to $2^n - 1$ are (α, β)-stochastic.*

2. There exists a constant c such that, for any n and any α and β with $2\alpha + \beta < n - 6\log n - c$ not all the numbers from 0 to $2^n - 1$ are (α, β)-stochastic.

Gács *et al.* [GTV01] improved Shen's result and proved that for every n there are objects of length n with complexity $C(x|n) \approx n$ such that every explicit algorithmic sufficient statistic for x has complexity about n.

Sophistication

Koppel [Kop88] used total functions to represent the useful information of infinite sequences and the resulting measure has been called *sophistication*. In this work we adapt Koppel's definition to finite strings.

Definition 2.6 ([Kop88]) *The sophistication of $x \in \Sigma^n$, with a significance level c, is defined by,*

$$soph_c(x) = \min_p\{|p| : p \text{ is total and there exists } d \text{ such that } U(p,d) = x \text{ and } |p| + |d| \leq K(x) + c\}$$

Computational Depth

The Kolmogorov complexity of a string x does not take into account the time necessary to produce the string from a description of length $C(x)$. Levin [Lev73], introduced a useful variant weighing program size and running time.

Definition 2.7 *For any strings x, y, the Levin complexity of x given y is*

$$Ct(x|y) \quad = \quad \min_p\{|p| + \log t : U(p,y) \text{ halts in at most } t \text{ steps and outputs } x\}.$$

After some attempts, Bennett [Ben88] formally defined the s-significant logical depth of a string x as the time required by a standard universal Turing machine to generate x by a program that is no more than s bits longer than the shortest descriptions of x. A string x is called logically deep if it takes a relatively large amount of time to generate it from any short description.

Definition 2.8 *Let x be a string and s be a nonnegative integer. The logical depth of x at a significance level s is*

$$depth_s(x) \quad = \quad \min_p\{t : U(p) \text{ halts and outputs } x \text{ in at most } t \text{ steps, and } |p| < C(x) + s\}$$

Note that algorithmically random strings are shallow at any significance level. In particular, Chaitin's Ω is shallow. Deep strings are hard to find, however they can be constructed by diagonalization, see [Ben88]. Bennett has proved that a fast deterministic processes is unable to transform a shallow object into a deep one, and that fast probabilistic processes can do so only with small probability (*slow growth law*).

Antunes, Fortnow, van Melkebeek and Vinodchandran [AFvMV06] consider logical depth as one instantiation of this more general theme and the authors propose several other variants and show many applications. Intuitively strings of high computational depth are low Kolmogorov complexity strings (and hence nonrandom), but a resource bounded machine cannot identify this fact. By considering $Ct(x) - C(x)$ we get a variation of Bennett's logical depth.

Definition 2.9 *For any string x the basic computational depth of x is*

$$\begin{aligned} bcd(x) &= Ct(x) - C(x) \\ &= \min_p\{\log(t) + |p| - C(x) : \\ &\qquad U(p) \text{ outputs } x \text{ in } t \text{ steps}\}. \end{aligned}$$

Basic computational depth has the advantage over logical depth that we do not need a significance level parameter. In fact, we are adding to (the logarithm of) the running time of the program a penalty $|p| - C(x)$ which can be viewed as a significance level.

Logically deep strings are not easy to identify, but can be constructed by diagonalization in time larger than 2^t for depth t [Ben88]. In [AFvMV06] the authors prove that there are an exponential number of strings with large basic computational depth. The result holds for Bennett's notion of logical depth as well.

Theorem 2.2 *There exists a constant c such that for any $0 < \epsilon < 1$ there are at least $2^{\epsilon n}$ strings x of length n satisfying*

$$bcd(x) \geq (1 - \epsilon)n - c \log n.$$

3 On the Existence of Highly Sophisticated Strings

Like depth, sophistication is a measure of the non-random information in a string. We start by recasting Definition 2.6 for finite strings.

Definition 3.1 *Let c be a constant, $x \in \Sigma^n$ and U the universal reference Turing machine*

$$soph_c(x) = \min\{|p| : p \text{ is total, exists } d\ [U(p, d) = x \text{ and } |p| + |d| \leq C(x) + c]\}.$$

Initial sequences of the halting set give a nice example of a language with high depth but low sophistication.

Example 3.1 *Let ϕ be a universal partial recursive function. Define $\chi = \chi_1\chi_2\chi_3...$ the characteristic sequence of the diagonal halting set $K_0 = \{i : \phi_i(i) < \infty\}$*

$$\chi_i = \begin{cases} 1 & if \quad i \in K_0 \\ 0 & otherwise \end{cases}$$

By the Barzdin's Lemma (see [LV97]) $\log n \leq C(\chi_{1:n}|n) \leq \log n + c'$, so $soph_c(\chi_{1:n}) \leq \log n + c'$, but $depth_s(\chi_{1:n})$ is very high ($\chi_{1:n}$ is very deep).

We could also have used the characteristic sequence of the recursive set constructed (by diagonalization) in Li and Vitányi [LV97, Theorem 7.1.4] to show this gap between logical depth and sophistication.

We now prove that there exists strings $x \in \Sigma^n$ such that the sophistication is close to n, i.e., highly sophisticated strings. This result answers the question raised by Kolmogorov about the existence of "absolutely non-random" (or absolutely non-stochastic) objects, Gács *et al.* [GTV01] proved a similar result independently.

Theorem 3.1 *Let c be a constant, for some string $x \in \Sigma^n$,*

$$soph_c(x) > n - 2\log n - 2c.$$

Proof. For all p such that $|p| \leq n - 2\log n - 2c$ we define

$$r_p = \begin{cases} 0 \text{ if there exists a } d \text{ such that } |d| < n - |p| - c \text{ and } U(p,d) \text{ diverges} \\ \max\limits_{d:|d|<n-|p|-c} \text{running time of } U(p,d) \end{cases}$$

Let $T = \max r_p$. Given n and p that maximizes r_p we can compute T. Consider

$$V = \{x : \exists p, d\ [|p| \leq n - 2\log n - 2c,\ |d| \leq n - |p| - c,\ U(p,d) = x \text{ in at most } T \text{ steps}]\}.$$

Let z be the least, in the lexicographic order, element in $\{0,1\}^n$ such that $z \notin V$. Such z exists since $\forall x \in V, C(x) \leq |p| + |d| \leq |p| + n - |p| - c = n - c$ and by a simple counting argument there must exist at least $2^n(1 - \frac{1}{2^c})$ strings $z \in \{0,1\}^n$ with $C(z) \geq n - c$. Since given n and p that maximizes r_p we can compute T, then by construction $C(z) \leq C(p) + 2\log n \leq n - 2c$.
Assume that $soph_c(z)$ is small, i.e., $soph_c(z) \leq n - 2\log n - 2c$; then, by definition there exist some total p^* and some d^* satisfying

$$|p^*| \leq n - 2\log n - 2c \text{ and } |p^*| + |d^*| \leq C(z) + c$$

but then we have that $|p^*| \leq n - 2\log n - 2c$ and $|d^*| \leq C(z) + c - |p^*| \leq n - |p^*| - c$ so $U(p^*, d^*)$ runs in time $\leq T$, i.e., $z \in V$. But by construction $z \notin V$, so

$$soph_c(z) > n - 2\log n - 2c.$$

$\diamond$

We can get a sharper result using conditional Kolmogorov complexity.

Corollary 3.1 *For some string x of length n, $soph_c(x|n) \geq n - 2c$.*

4 Coarse Sophistication

Koppel's definition of sophistication, Definition 3.1, may not be stable. Small changes in c could cause large changes in $\text{soph}_c(x)$. In this section, we consider a new notion of *coarse sophistication* that incorporate the "constant" as a penalty in the formula to obtain a more robust measure.

Definition 4.1 *The coarse sophistication of a string $x \in \Sigma^n$, is defined as*

$$csoph(x) = \min\{2|p| + |d| - C(x) : U(p,d) = x \text{ and } p \text{ is total.}\}$$

Think of this definition as $|p|$ for sophistication plus a penalty $|p| + |d| - C(x)$ for how far away we are from the minimal program. The choice of using $|p| + |d| - C(x)$ instead of some other penalty function is admittedly arbitrary but it seems natural and does lead to some interesting consequences.

Some "sensibility" is lost as now $csoph(x)$ is upper bounded by $\frac{|x|}{2}$.

Theorem 4.1 *There is a constant c such that for all $x \in \Sigma^n$, $csoph(x) \leq \frac{n}{2} + c$.*

Proof. If $C(x) \leq \frac{n}{2}$ then by definition $csoph(x) \leq \frac{n}{2} + c$.
If $C(x) > \frac{n}{2}$ then considering the $print(x)$ program we have $csoph(x) \leq \frac{n}{2} + c$. $\diamond$

There are strings for which this upper bound is tight.

Theorem 4.2 *For some string x of length n, $csoph(x) > \frac{n}{2} - 4\log n$.*

Proof. For all p such that $|p| \leq n - 2\log n$ we define

$$r_p = \begin{cases} 0 \text{ if exists } d : |d| < n - |p| \text{ such that } U(p,d) \text{ diverges} \\ \max\limits_{d:|d|<n-|p|} \text{running time of } U(p,d) \end{cases}$$

Let $S = \max r_p$. Given n and p that maximizes r_p we can compute S. Consider

$$V = \{x : \text{ exists } p, d \ [|p| \leq \frac{n}{2} - 2\log n, \ |d| \leq n - 2|p| - 2\log n, \ U(p,d) = x \text{ in at most } S \text{ steps}]\}.$$

Let z be the least, in the lexicographic order, element in $\{0,1\}^n$ such that $z \notin V$. Such z exists since for all $x \in V, C(x) < |p| + |d| + 2\log n \leq |p| + 2\log n + n - 2|p| - 2\log n = n - |p| < n$ and by a simple counting argument there exists random strings. By construction we know that

$$C(z) \leq C(p) + 2\log n \leq \frac{n}{2}.$$

Assume that $csoph(z)$ is small, i.e., $csoph(z) \leq \frac{n}{2} - 4\log n$; the program p^* which defines the sophistication is such that

$$|p^*| \leq \frac{n}{2} - 2\log n$$

and there exists d^* such that

$$2|p^*| + |d^*| - C(z) \leq \frac{n}{2} - 2\log n$$

but then we have that

$$|d^*| \leq \frac{n}{2} - 2\log n + C(z) - 2|p^*| \leq n - 2\log n - 2|p^*|$$

so $U(p^*, d^*)$ runs in time $\leq S$, i.e., $z \in V$. But by construction $z \notin V$, so

$$csoph(z) > \frac{n}{2} - 4\log n.$$

◇

We investigate the computational power of coarse sophistication, relating coarse sophistication with the halting problem. We prove that with the knowledge of the sophistication of a string and some extra $\log n$ bits we can solve the halting problem for all programs of length smaller than $\frac{csoph(x)}{2} - 2\log n$.

Theorem 4.3 *For all $x \in \Sigma^n$, given x and $O(\log n)$ bits we can solve the halting problem for all programs q such that $|q| < \frac{csoph(x)}{2} - 2\log n$.*

Proof. With the given $O(\log n)$ bits find the minimum program p such that $U(p) = x$, and consider S its running time. Suppose that there is some q such that $|q| < \frac{csoph(x)}{2} - 2\log n$ and $U(q)$ converges in time $v > S$. Consider the program w such that $U(w,p)$ first computes v and then simulates $U(p)$ for v steps, producing x. Now w is total and there is a constant c such that $|w| = |q| + c$, and

$$\begin{aligned} csoph(x) &\leq 2|w| + |p| - C(x) \\ &\leq 2|q| + 2c + |p| - C(x) \\ &\leq csoph(x) - 2\log n + 2c + |p| - C(x) \\ &< csoph(x) - 2\log n + 2c \end{aligned}$$

so such an input can not exist. ◇

5 Coarse Sophistication vs Busy Beaver Computational Depth

The main drawback of basic computational depth is the fact that it is only suitable for strings whose program runs in time at most exponential in the length of the string. This motivate the definition of *busy beaver computational depth*, preserving the intuition of basic computational depth and capturing all possible running times. Besides, using properly the Busy Beaver function, we can scale down computational depth from running time to program length. Often the busy beaver function, $BB(n)$, is defined to be the largest number which can be computed by an n-state Turing machine. Although many variations on the definition of busy beaver have been used, we recast the original definition (see Daley [Dal82]) as follows:

Definition 5.1 *The* busy beaver *function is defined* $BB : \mathbf{N} \to \mathbf{N}$ *as*

$$BB(n) = \max_{p:|p|\leq n} \{\textit{Running time of } U(p) \textit{ when defined}\}$$

Definition 5.2 *The busy beaver computational depth of* $x \in \Sigma^n$*, is defined as*

$$depth_{bb}(x) = \min\{|p| - C(x) + k : U(p) = x \textit{ in } t \textit{ steps and } t \leq BB(k)\}$$

As in basic computational depth, $depth_{bb}$ also incorporate a significance level in the formula; $|p| - C(x)$ is a penalty measuring how far away we are from the minimal program. It is important to note that, instead of using the running time, $depth_{bb}$ uses the inverse Busy Beaver of the running time. However, as for coarse sophistication, some "sensibility" is lost, $depth_{bb}$ is nearly upper bounded by $\frac{n}{2}$.

Theorem 5.1 *There is a constant* c *such that for all* $x \in \Sigma^n$, $depth_{bb}(x) \leq \frac{n}{2} + BB^{-1}(n) + c$.

Proof. If $C(x) \leq \frac{n}{2}$, considering the minimum program producing x we have $depth_{bb}(x) \leq \frac{n}{2}$.
If $C(x) > \frac{n}{2}$, considering the $print(x)$ program we have $depth_{bb}(x) \leq \frac{n}{2} + BB^{-1}(n) + c$. ◇

There are strings for which this upper bound is tight.

Theorem 5.2 *For some string* x *of length* n, $depth_{bb}(x) > \frac{n}{2} - 2\log n$

Proof. Let V be the set of all $x \in \Sigma^n$ such that there exists p and k such that $|p| \leq \frac{n}{2} - 2\log n$, $k \leq n - 2\log n - |p|$, $U(p) = x$ in t steps and $t \leq BB(k)$.

Consider z that is the least, in the lexicographic order, element in Σ^n such that $z \notin V$. Such z exists since for all $x \in V, C(x) < |p| + 2\log n \leq \frac{n}{2}$. We can approximate BB from below so V is r.e. and by construction we know that the size of V is smaller than $2^{\frac{n}{2} - 2\log n}$ so $C(z) < \frac{n}{2}$.
Assume $depth_{bb}(z) \leq \frac{n}{2} - 2\log n$, then by definition exists a p and k such that

$$k + |p| - C(z) \leq \frac{n}{2} - 2\log n \text{ and } U(p) = x \text{ in } t \text{ steps and } t \leq BB(k),$$

i.e.,

$$k \leq n - 2\log n - |p| \text{ and } U(p) = x \text{ in } t \text{ steps and } t \leq BB(k)$$

proving that $z \in V$. But by construction $z \notin V$, so $depth_{bb}(z) > \frac{n}{2} - 2\log n$. ◇

We now prove the equivalence between coarse sophistication and busy beaver computational depth.

Theorem 5.3 *For all* $x \in \Sigma^n$, $|csoph(x) - depth_{bb}(x)| \leq O(\log n)$.

Proof. We use p_s to denote the program associated with $csoph$ and p_d with $depth_{bb}$.

- We start proving that $depth_{bb}(x) \leq csoph(x) + O(\log n)$.
  For all d such that $|d| \leq n$, let t be the maximum running time of $U(p_s, d)$, then using p_s and $\log n$ bits to describe d we get $t \leq BB(|p_s| + O(\log n))$. So we get an upper bound on the depth of x

$$\begin{aligned} depth_{bb}(x) &\leq |p_s| + |d| - C(x) + |p_s| + O(\log n) \\ &\leq csoph(x) + O(\log n) \end{aligned}$$

- Now we prove that $csoph(x) \leq depth_{bb}(x) + O(\log n)$.
  We denote the running time of a program p by $rt(p)$. Let q be the first program of length $k = BB^{-1}(rt(p_d))$ that satisfies

$$|q| = k,\ rt(q) \geq rt(p_d) \text{ and for all } u : |u| = k,\ rt(u) > rt(p_d) \Rightarrow rt(q) \leq rt(u)$$

  that is, the running time of q immediately follows the running time of p_d. Consider the set

$$A = \{v : |v| = |p_d|,\ rt(v) < rt(q) \text{ and for all } u : |u| = k,\ rt(u) > rt(v) \Rightarrow rt(u) > rt(q)\}.$$

  Given $q, n, |p_d|$ and k, A is recursive since $rt(q)$ is used as the time limit for the running time of all programs. By symmetry of information we have

$$C(v|q) \leq C(q|v) + C(v) - C(q) + O(\log n).$$

  But, given v we can use its running time to get q, because q is the first program of length k whose running time immediately follows the running time of p_d, so $C(q|v) \leq O(\log n)$ and $C(v|q) \leq |p_d| - k + O(\log n)$.
  As A is recursive we have $|A| \leq 2^{|p_d|-k+O(\log n)}$ and we can point p_d with its index i in A. Note that for every $p \in A$, $rt(p) < rt(q)$, i.e., it halts. Now consider the code of the machine that with input $< q, n, |p_d|, k >$ and i constructs the set A and picks p_d that given as input to the universal Turing machine produces x. Then

$$\begin{aligned} csoph(x) &\leq 2|q| + O(\log n) + |i| - C(x) \\ &\leq 2k + O(\log n) + |p_d| - k - C(x) \\ &\leq k + |p_d| - C(x) + O(\log n) \\ &\leq depth_{bb}(x) + O(\log n) \end{aligned}$$

◇

Acknowledgment

The authors thanks Paul Vitányi and the anonymous reviewers for their comments.

References

[Ant02] L. Antunes. Useful Information. *PhD Thesis*, Computer Science Department, Oporto University, 2002.

[AF03] L. Antunes and L. Fortnow. Sophistication Revisited. In *Proceedings of the Proceedings of the 30th International Colloquium on Automata, Languages and Programming*, volume 2719 of Lecture Notes in Computer Science, pages 267-277. Springer, 2003.

[AFvMV06] L. Antunes, L. Fortnow, D. van Melkebeek, and N. Vinodchandran. Computational depth: Concept and applications. *Theoretical Computer Science*, 354(3):391-404, April 2006.

[Ben88] C. H. Bennett. Logical depth and physical complexity. In R. Herken, editor, *The Universal Turing Machine: A Half-Century Survey*, pages 227–257. Oxford University Press, 1988.

[Cha66] G. Chaitin. On the length of programs for computing finite binary sequences. *Journal of the ACM.* 13:4(1966), 145-149.

[Cov85] T. M. Cover. Kolmogorov Complexity, Data Compression, and Inference, pp. 23-33. In *The Impact of Processing Techniques on Communications.* J. K. Skwirzynski, Ed., Martinus Nijhoff Publishers, 1985.

[Dal82] R. Daley. Busy Beaver Sets: Characterizations and Applications. *In Information and Control*, 52(1982), 52-67.

[GTV01] P. Gács, John Tromp and P. Vitányi. Algorithmic Statistics. *In IEEE Trans. Inform. Theory,* 47:6(2001), 2443-2463.

[Kol65] A. N. Kolmogorov. Three approaches to the quantitative definition of information. *Problemy Inform. Transmission*, 1:1(1965), 1-7.

[Kop88] M. Koppel. Structure, pp. 435-452. In *The Universal Turing Machine: A Half-Century Survey.* R. Herken, Ed., Oxford University Press, 1988.

[Kop91] M. Koppel. Learning to Predict Non-Deterministically Generated Strings. *Machine Learning*, 7(1991), 85-99.

[KA91] M. Koppel and H. Atlan. An Almost Machine-Independent Theory of Program-Length Complexity, Sophistication, and Induction. *Information Sciences*, 56(1991), 23-33.

[Lev73] L. Levin. Universal Search Problems. *Problems Inform. Transmission*, 9(1973), 265-266.

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

[LV97] M. Li and P. Vitányi. *An introduction to Kolmogorov complexity and its applications.* Springer, 2nd edition, 1997.

[Sch73] C. P. Schnorr. Process Complexity and Effective Random Tests. *J. Comp. System Sci.*, 7(1973), 376-388.

[She83] A. Shen. The Concept of (α, β)-Stochasticity in the Kolmogorov Sense, and its Properties. *Soviet Math. Dokl.*, 28(1983), 295-299.

[Sol64] R. Solomonoff. A Formal Theory of Inductive Inference, Part I. *Information and Control*, 7:1(1964), 1-22.

[VV02] N. Vereshchagin and P. Vitányi. Kolmogorov's structure functions and an application to the foundations of model selection. *Proc. 47th IEEE Symp. Found. Comput. Sci.*, 2002.

[Vit06] P. Vitányi. Meaningful information. *IEEE Transactions on Information Theory.* 52:10(2006),4617-4627.

[V'y99] V.V. V'yugin. Algorithmic complexity and stochastic properties of finite binary sequences. *The Computer Journal*, 42(4):294-317, 1999.