Kolmogorov Complexity and the Incompressibility Method

Holger Arnold*

1. Introduction. What makes one object more complex than another? Kolmogorov complexity, or program-size complexity, provides one of many possible answers to this fundamental question. In this theory, whose foundations have been developed independently by R. J. Solomonoff [16–18], A. N. Kolmogorov [7, 8], and G. Chaitin [4] in the 1960s, the complexity of an object is defined as the length of its shortest effective description, which is the minimum number of symbols that must be specified such that the object can be reproduced from the specification by some abstract computing machine or formal system.

The minimal length for an effective description of an object obviously depends on the exact method used for reproducing the object from the description. Since we want to measure the complexity of objects independently of any particular model of computation, we therefore define Kolmogorov complexity relative to a fixed (but unspecified) universal computable function. Then, using any model of computation, objects can be described at most a constant number of symbols shorter than their Kolmogorov complexity, where 'constant' refers to a value that depends on the method for reproducing the objects but not on the described objects themselves. We can ignore these additive constants because we are typically interested in asymptotic complexities rather than in numeric complexity values.

The notion of Kolmogorov complexity yields a simple yet powerful proof technique, called the incompressibility method. The purpose of this paper is to explain the concepts on which this technique is based and, along the way, to provide a concise introduction to Kolmogorov complexity theory. The only prerequisite is a basic understanding of computable functions; the employed notation is summarized in a short paragraph after the main text. The definitions and theorems in this paper are more or less standard; if not stated otherwise, they have been taken (with some modifications) from the book by Li and Vitányi [9], which is a comprehensive reference on Kolmogorov complexity theory and it applications.

2. Descriptions and complexity. First we need to make precise what we mean by a description of an object and its complexity. In the context of Kolmogorov complexity, an 'object' is a finite string over some fixed finite alphabet. The complexity of more abstract objects, such as numbers or sets, can therefore only be measured relative to a particular encoding of these objects as strings. In this paper we assume that all strings are built

Date: October 24, 2011.

*See `http://harnold.org/` for contact information.

from the symbols '0' and '1', but this assumption is not essential; all presented results hold similarly for strings over other alphabets.

Definition 1. A *description* of a string x consists of a partial computable function φ and strings p and y such that $\varphi(p, y) = x$. Such a p is called a *program* for computing x by φ given y. The *φ-complexity* C_φ of x relative to y is defined as

$$C_\varphi(x \,|\, y) = \min \{\, l(p) \mid p \in \mathbf{B}^* \text{ and } \varphi(p, y) = x \,\}$$

if there is a program p for computing x by φ given y, and as $C_\varphi(x \,|\, y) = \infty$ otherwise.

This definition allows us to describe objects using any model of computation whose computational effects can be expressed by partial computable functions. Since the functions we are using in this paper are sufficiently simple, we will define them by equations comprising only constructs whose computability is self-evident. When the description of an object requires more complex functions, however, it is better to specify these functions using a particular model of computation.

Up to additive constants, the notion of complexity introduced in Definition 1 is invariant with respect to the chosen description method, which is shown by the following theorem. Its proof relies on the existence of a universal partial computable function [13, §1.8], a fundamental result proved by A. Turing [19]:

Theorem 2 (Invariance Theorem)**.** *There exists a partial computable function φ_0 such that $C_{\varphi_0}(x \,|\, y) \leq C_\varphi(x \,|\, y) + c_\varphi$ for all partial computable functions φ and all strings x and y, where c_φ is a nonnegative integer depending on φ but not on x or y.*

Proof. Let u be a universal partial computable function such that $u(n, x) = \psi_n(x)$ for all nonnegative integers n and all strings x, where $\psi_0, \psi_1, \ldots$ is an enumeration of the partial computable functions, and define φ_0 by $\varphi_0(\langle n, p\rangle, y) = u(n, \langle p, y\rangle)$ for all nonnegative integers n and all strings p and y.[1] Now let φ be any partial computable function; then $\varphi = \psi_n$ for some nonnegative integer n. If $C_\varphi(x \,|\, y) < \infty$ for strings x and y, then $x = \varphi(p, y) = u(n, \langle p, y\rangle) = \varphi_0(\langle n, p\rangle, y)$ for some string p with $l(p) = C_\varphi(x \,|\, y)$. Hence $C_{\varphi_0}(x \,|\, y) \leq l(\langle n, p\rangle) = l(p) + 2l(n) + 1 = C_\varphi(x \,|\, y) + 2l(n) + 1$ (a shorter encoding of n and p is possible, but this is not important here). □

A partial computable function with the property specified in Theorem 2 is called an *additively optimal universal description function*. The invariance theorem justifies choosing a fixed such function φ_0, which we will call the *reference function*, and defining Kolmogorov complexity relative to this function:

[1] To simplify the definitions of partial computable functions, we implicitly assume that a function is undefined for arguments not matching any of the given definitions. For example, the function φ_0 in the proof of Theorem 2 is undefined if its argument does not have the form $\langle\langle u, v\rangle, w\rangle$ for strings u, v, and w.

Definition 3. Let φ_0 be a fixed additively optimal universal description function such that $C_{\varphi_0}(x \mid y) > 0$ for all strings x and y.[2] Then, for all strings x and y, the φ_0-complexity $C_{\varphi_0}(x \mid y)$ is called the *Kolmogorov complexity* of x relative to y, denoted by $C(x \mid y)$, and $C(x \mid \epsilon)$ is called the Kolmogorov complexity of x, denoted by $C(x)$.

This is the "classical" definition of Kolmogorov complexity. There are other complexity measures that are superior to this definition in some respects. In particular, if we require that no valid program should be a prefix of another valid program, we get a complexity measure, called *prefix complexity*, that has a number of theoretical advantages over the classical definition. In G. Chaitin's opinion, the original definition of Kolmogorov complexity is therefore "*only of historical or pedagogic interest*" [3, Foreword]. On the other hand, the classical definition is often easier to use in calculations because it admits a simple, computable upper bound on the complexity of a string, namely its length plus a constant (Lemma 4), whereas the corresponding upper bound on the prefix complexity of a string necessarily involves the complexity of the string's length.

It is easy to show that Kolmogorov complexity is not computable: if C were a partial computable function, then, by applying Kleene's recursion theorem [13, §11.2], we could construct a program of length n producing a string of complexity greater than n—a contradiction resembling Berry's paradox of "*the least integer not nameable in fewer than nineteen syllables*" [14]. Nevertheless, the Kolmogorov complexity function can be effectively approximated from above using dovetailing.

Since it implicitly depends on the chosen reference function—and there are infinitely many additively optimal universal description functions—Kolmogorov complexity is not an absolute notion of complexity for individual strings. In fact, by choosing a suitable reference function, we can assign nearly arbitrary complexity values to any finite subset of strings. By the invariance theorem, however, the effect of such "distortions" of the complexity function diminishes with increasing complexity, as Kolmogorov already observed: "*It is true, as we have already noted, that such an individual quantitative estimate of information is meaningful only when the quantity of information is sufficiently large*" (Kolmogorov identified complexity with information) [8, Section 3]. Therefore, to obtain results that are independent of the chosen reference function, we have to analyze not the complexity of individual strings but the asymptotic complexity of infinite families of strings of increasing size.

3. Complexity upper bounds. By the invariance theorem (Theorem 2), every program for computing a string by some partial computable function implies an upper bound on the complexity of the string. We can use this fact to derive a number of simple but important upper bounds.

Lemma 4. *There is a nonnegative integer c such that $C(x) \leq l(x) + c$ for all strings x.*

[2] This assumption is only made to simplify some calculations involving logarithms of complexities.

Proof. Let f be the partial computable function defined by $f(x, y) = x$ for all strings x and y. Then, by Theorem 2, there is a nonnegative integer c such that $C(x) \leq C_f(x)+c = l(x) + c$ for all strings x. □

Lemma 5. *Let t be a computable function. Then there is a nonnegative integer c_t such that $C(t(x) \,|\, y) \leq C(x \,|\, y) + c_t$ for all strings x and y.*

Proof. Let f be the partial computable function defined by $f(p, y) = t(\varphi_0(p, y))$ for all strings p and y. Then every program p for computing x by φ_0 given y is also a program for computing $t(x)$ by f given y. Hence, by Theorem 2, there is a nonnegative integer c_t such that $C(t(x) \,|\, y) \leq C_f(t(x) \,|\, y) + c_t \leq C_{\varphi_0}(x \,|\, y) + c_t = C(x \,|\, y) + c_t$. □

Lemma 6. *Let t be a computable function. Then there is a nonnegative integer c_t such that $C(x \,|\, y) \leq C(x \,|\, t(y)) + c_t$ for all strings x and y.*

Proof. Let f be the partial computable function defined by $f(p, y) = \varphi_0(p, t(y))$ for all strings p and y. Then every program p for computing x by φ_0 given $t(y)$ is also a program for computing x by f given y. Hence, by Theorem 2, there is a nonnegative integer c_t such that $C(x \,|\, y) \leq C_f(x \,|\, y) + c_t \leq C_{\varphi_0}(x \,|\, t(y)) + c_t = C(x \,|\, t(y)) + c_t$. □

Lemma 6 implies that there is a nonnegative integer c such that $C(x \,|\, y) \leq C(x) + c$ for all strings x and y (let t to be the constant function ϵ), and that for every string y, there is a nonnegative integer c_y such that $C(x) \leq C(x \,|\, y) + c_y$ for all strings x (let t be a computable function mapping ϵ to y).

Lemma 7. *There is a nonnegative integer c such that $C(\langle x, y\rangle) \leq C(x) + C(y) + 2 \log \min \{C(x), C(y)\} + c$ for all strings x and y.*

Proof. Let f be the partial computable function defined, for all strings p, q, and z, by $f(\overline{l(p)}pq, z) = \langle \varphi_0(p, \epsilon), \varphi_0(q, \epsilon)\rangle$. Then, for every program p for computing x by φ_0 and every program q for computing y by φ_0, the string $\overline{l(p)}pq$ is a program for computing $\langle x, y\rangle$ by f, with $l(\overline{l(p)}pq) = l(p) + l(q) + 2\lfloor \log (l(p) + 1)\rfloor + 1 \leq l(p) + l(q) + 2 \log l(p) + 2$. Hence, by Theorem 2, there is a nonnegative integer c such that $C(\langle x, y\rangle) \leq C_f(\langle x, y\rangle) + c \leq C(x) + C(y) + 2 \log C(x) + c + 2$ for all strings x and y. The lemma then follows by reasoning analogously for the function g defined, for all strings p, q, and z, by $g(\overline{l(p)}pq, z) = \langle \varphi_0(q, \epsilon), \varphi_0(p, \epsilon)\rangle$. □

For every injective pairing function p, it can be shown that there is a nonnegative integer c_p such that, for all nonnegative integers n, there are strings x and y of length at most n with $C(p(x, y)) \geq C(x) + C(y) + \log n - c_p$.[3] Thus, the logarithmic term in the bound of Lemma 7 can not be eliminated by choosing a different encoding of pairs.

[3] See the discussion of Examples 2.1.5 and 2.2.3 in Li and Vitányi [9].

4. Incompressible strings and randomness. A string is called compressible if it has a description which is shorter than the string itself. It is clear that some strings can be compressed by large amounts; in fact, the ratio of string length to string complexity can grow as fast as any computable function. For the development of Kolmogorov complexity theory, however, the interesting objects are the strings that can *not* be compressed:

Definition 8. A string x is called *k-incompressible* for some nonnegative integer k if $C(x) \geq l(x) - k$. A 0-incompressible string is called *incompressible*.

Incompressible strings lack regularities that could be exploited to obtain a compressed description for them; they are effectively patternless. It seems reasonable to call such strings *random*, at least insofar as this term is meaningful for finite objects. The intuitive correspondence between incompressibility and randomness is supported by a formal argument: an effective randomness test, as defined in Per Martin-Löf's theory of randomness [10], cannot distinguish incompressible strings from "truly random" strings if their length exceeds a constant depending on the test; in other words, all incompressible strings whose length is greater than this constant pass the test.

The following lemma provides a lower bound on the number of k-incompressible strings of a given length. It generalizes a bound that Buhrman et al. [2, Lemma 2] have proved for the case $k = 0$. In particular, the lemma shows that, as k increases, the frequency of k-incompressible strings in the set of all strings of length n converges rapidly to 1:

Lemma 9 (Incompressibility Lemma)**.** *There is a real number d with $0 \leq d < 1$ and a nonnegative integer c such that, for all nonnegative integers k, all integers $n \geq k + c$, and all strings y, there are at least $2^n(1 - 2^{-k}d)$ strings x of length n with $C(x \mid y) \geq n - k$.*

Proof. By Lemmas 4 and 6, there is a nonnegative integer c such that $C(x \mid y) \leq l(x) - c$ for all strings x and y. This means that, for all nonnegative integers n and k with $n \geq k + c$ and all strings y, at least $2^{n-k-c} - 1$ strings of length less than $n - k$ are programs for computing, by φ_0 and given y, strings of length less than $n - k$. Therefore at most $2^{n-k} - 2^{n-k-c}$ strings of length less than $n - k$ are programs for computing strings of length at least n. Hence there are at least $2^n(1 - 2^{-k}(1 - 2^{-c}))$ strings x of length n with $C(x \mid y) \geq n - k$. □

Although Lemma 9 shows that, already for small values of k, nearly all strings of every length are k-incompressible, it is impossible to prove the incompressibility of any particular string, except for very short ones. This is a consequence of a theorem proved by Chaitin [5], which states that there is a constant c such that no sound formal system of complexity n can prove that any particular string is of complexity greater than $n + c$. Although this theorem, which is known as Chaitin's incompleteness theorem, has similar implications as Gödel's first incompleteness theorem [6], its philosophical relevance has

occasionally been overrated by Chaitin himself and others; see van Lambalgen [20] and Raatikainen [12] for a critical discussion of some of Chaitin's arguments.

5. Complexity of strings in finite sets. In many applications of Kolmogorov complexity, objects are not considered in isolation but as members of certain finite sets, which requires a way of encoding finite sets of strings as strings. For this purpose, we call a surjective function σ mapping strings to finite subsets of $\mathbf{B}^*$ a *representation scheme* for finite sets of strings if, for all finite sets A of strings, membership in A and the number of elements in A are computable from any string x with $\sigma(x) = A$. For example, we can define σ so that it maps every string x to the subset of $\mathbf{B}^*$ whose characteristic sequence, when generated from the shortlex order on $\mathbf{B}^*$, is $x000\ldots$.[4] Given a representation scheme σ, we will refer to every string x with $\sigma(x) = A$ as a *representation* of the set A. In the following sections, σ denotes any representation scheme for finite sets of strings; we must keep in mind, however, that the derived results depend quantitatively on the chosen scheme.

The elements of a finite set of strings can be computed from a representation of the set and their index relative to some linear order on the set elements. Thus, their complexity relative to any representation of the set is bounded from above by the logarithm of the set's cardinality plus a constant:

Lemma 10. *There is a nonnegative integer c such that $C(x \mid y) \leq \lfloor \log |A| \rfloor + c$ for all finite sets A of strings, all representations y of A, and all $x \in A$.*

Proof. Let f be the partial computable function defined by $f(i, y) = x_i$ for all strings y and all nonnegative integers $i \leq |\sigma(y)|$, where x_i is the i-th element of the set $\sigma(y)$ relative to some computable linear order on $\mathbf{B}^*$. The elements of $\sigma(y)$ are computable because σ is a representation scheme. Then, by Theorem 2, there is a constant c such that $C(x \mid y) \leq C_f(x \mid y) + c \leq \lfloor \log |A| \rfloor + c$ for all finite sets A of strings, all representations y of A, and all $x \in A$. □

In Section 4 we used the length of a string as a natural upper bound on its complexity, from which we derived the notion of k-incompressibility. Similarly, we can use $\lfloor \log n \rfloor$ as a natural upper bound on the complexity of strings contained in a finite set of n strings, relative to a representation of the set:

Definition 11. Let y be a string, and let $A = \sigma(y)$. Then a string $x \in A$ is called *k-incompressible* relative to y for some nonnegative integer k if $C(x \mid y) \geq \lfloor \log |A| \rfloor - k$.

The following lemma, which is also called incompressibility lemma, provides a lower bound on the number of k-incompressible strings in any finite set of strings. Note that, without making assumptions about the strings in the set, the best we can do is to simply

[4] This representation scheme is used by Li and Vitányi [9] (Exercise: Find this definition in their book).

count *all* potential programs. As a result, the bound in this lemma is weaker than the bound in Lemma 9:

Lemma 12 (Incompressibility Lemma)**.** *Let A be a finite set of strings. Then, for all nonnegative integers $k \leq \log |A|$ and all strings y, the set A contains at least $(1-2^{-k})\,|A|+1$ strings x with $C(x \,|\, y) \geq \lfloor \log |A| \rfloor - k$.*

Proof. There are $\sum_{0 \leq i \leq k-1} 2^i = 2^k - 1$ strings of length less than k. Hence A contains at most $2^{-k}\,|A| - 1$ strings of complexity less than $\lfloor \log |A| \rfloor - k$. □

The second incompressibility lemma shows that most strings in any finite set of strings cannot be described much shorter than by their index in the set. In other words, a *typical* element of a finite set of strings has close to maximal complexity relative to the set. We can quantify the degree of typicality of a string relative to a finite set of strings by considering the distance of the string's complexity, relative to a representation of the set, to the maximal complexity for elements of the set:

Definition 13. Let y be a string, and let $A = \sigma(y)$. Then, for all strings x, the *randomness deficiency* $\delta(x \,|\, y)$ of x relative to y is defined as $\delta(x \,|\, y) = \lfloor \log |A| \rfloor - C(x \,|\, y)$ if $x \in A$, and as $\delta(x \,|\, y) = \infty$ if $x \notin A$.

By Lemma 10, there is a nonnegative integer c such that $\delta(x \,|\, y) > -c$ for all strings x and y, and from Lemma 12, it follows immediately that any finite set A of strings with representation y contains at most $2^{-k}\,|A| - 1$ strings x with $\delta(x \,|\, y) > k$, for all nonnegative integers $k \leq \log |A|$.

6. The incompressibility method. The main result that we derived from the concepts introduced in the preceding sections is that, in every finite set of strings, only very few elements can be compressed by large amounts. That this result can be exploited in a proof has been shown first by W. Paul [11], who proved a new lower bound on the running time of sorting algorithms on multi-tape Turing machines. Since then, arguments using the properties of incompressible strings have found many applications; in particular, they have been used to prove lower bounds on the time complexity of computational problems, to characterize the average-case behavior of algorithms, and to prove the existence or high probability of combinatorial objects with certain properties. See Li and Vitányi's book [9] for a survey of many of these applications, including extensive references to original research in this field.

The name "incompressibility method" subsumes different proof techniques that are based on the properties of incompressible strings. In this paper we consider only one such technique: the replacement of probabilistic arguments for proving the existence of combinatorial objects with certain properties. Using the probabilistic method [1], we can prove that a given domain contains an object with a given property by showing that an object chosen randomly from the domain has this property with positive probability.

Since we know from Lemmas 9 and 12 that every domain contains incompressible objects, we can replace such a probabilistic existence argument by a non-probabilistic one if we can show that every incompressible object in the domain has the desired property, and the usual way to show this is to demonstrate that every object that does not have this property can be compressed.

Proofs based on incompressibility arguments are often simpler than proofs based on purely probabilistic arguments. One reason for this apparent simplicity is that when using incompressibility arguments, reasoning can be confined to a well-defined subset of objects—the incompressible elements in a domain. In contrast, when using probabilistic arguments, it is necessary to consider a whole distribution of objects. Moreover, since, frequently, the very definition of a property already indicates a method for compressing objects that do not have this property, incompressibility arguments are usually very intuitive.

For finite domains (and infinite families of finite domains) and computable properties, incompressibility arguments and probabilistic arguments are equivalent in the following sense: If all elements with sufficiently small randomness deficiency have a given property, then elements chosen randomly according to the uniform distribution have this property with high probability.[5] Conversely, elements with sufficiently small randomness deficiency share all properties that randomly chosen elements have with high probability. The following theorem makes these statements precise:

Lemma 14. *Let A be a finite set of strings, and let y be a representation of A. Let d be an integer such that $d \leq \log |A|$. Let P be a computable predicate.*

1. *If all strings $x \in A$ with $\delta(x \,|\, y) \leq d$ satisfy P, then at least $(1 - 2^{-d})\,|A| + 1$ strings in A satisfy P.*

2. *There is a nonnegative integer c_P, not depending on A, such that, if at least $(1 - 2^{-d})\,|A|$ strings in A satisfy P, then all $x \in A$ with $\delta(x \,|\, y) < d - c_P$ satisfy P.*

Proof. First, assume that all strings $x \in A$ with $\delta(x \,|\, y) \leq d$ satisfy P. Then, by Lemma 12, $|\{\, x \in A \mid P(x) \,\}| \geq |\{\, x \in A \mid \delta(x \,|\, y) \leq d \,\}| \geq (1 - 2^{-d})\,|A| + 1$.

To prove the second statement, assume that at least $(1-2^{-d})\,|A|$ strings in A satisfy P. Let f be the function defined by $\sigma(f(x)) = \{\, x \in \sigma(x) \mid \neg P(x) \,\}$ for all strings x. This function is computable because σ is a representation scheme and P is a computable predicate. Hence, by Lemma 6, there is a nonnegative integer c_f, depending on f (and thus on P) but not on A, such that $C(x \,|\, y) \leq C(x \,|\, f(y)) + c_f$ for all strings x. Let $B = \sigma(f(y)) = \{\, x \in A \mid \neg P(x) \,\}$. Then, by Lemma 10, there is a nonnegative integer c, not depending on A, B, or P, such that $C(x \,|\, f(y)) \leq \lfloor \log |B| \rfloor + c \leq \lfloor \log 2^{-d}\,|A| \rfloor + c \leq$

[5] Similar results can be obtained for more general classes of distributions, but it is necessary to preclude distributions that are too unbalanced in the sense that they assign the greatest part of the probability mass to the few highly compressible objects.

$\lfloor \log |A| \rfloor - d + c$ for all $x \in B$. Hence $\delta(x \,|\, y) \geq d - c_f - c$ for all $x \in A$ that do not satisfy P. □

Acknowledgment. I wish to thank Christoph Kreitz and Tim Richter for their comments on the first draft of this paper.

Notation. $\mathbf{N}$ denotes the set of nonnegative integers, $\mathbf{B}$ the set $\{0, 1\}$, $\mathbf{B}^n$ the set of binary strings of length n, and $\mathbf{B}^*$ the set of finite binary strings. The term 'string' implicitly refers to a finite binary string. For a string x, $l(x)$ denotes its length, and $\overline{x} = 0^{l(x)}1x$ denotes a self-delimiting encoding of x with $l(\overline{x}) = 2l(x) + 1$. String concatenation is denoted by juxtaposition. We identify binary strings and nonnegative integers by mapping strings bijectively to their index in $\mathbf{B}^*$ with respect to the shortlex order[6], starting with the empty string ϵ, which is mapped to 0. For a nonnegative integer n, $l(n) = \lfloor \log(n+1) \rfloor$ denotes the length of its representation as a binary string; $\log n$ denotes the binary logarithm of n. If not stated otherwise, 'constants' are nonnegative integers. $|S|$ denotes the cardinality of the set S. Partial computable functions map binary strings to binary strings. We use the term 'computable function' to refer to an everywhere defined partial computable function. A 'computable predicate' is a computable function with range $\{0, 1\}$, where 0 corresponds to the value *false* and 1 corresponds to the value *true*. For all positive integers n, $\langle \cdot \rangle^n$ denotes the injective map from $(\mathbf{B}^*)^n$ to $\mathbf{B}^*$ defined by $\langle x \rangle = x$, $\langle x_1, x_2 \rangle = \overline{x}_1 x_2$, and $\langle x_1, \ldots, x_n \rangle = \langle x_1, \langle x_2, \ldots, x_n \rangle \rangle$ for $n > 2$ (we omit the upper index when the number of arguments is clear). For all partial computable functions φ, we define $\varphi(x_1, \ldots, x_n) = \varphi(\langle x_1, \ldots, x_n \rangle)$.

References

1. Noga Alon and Joel H. Spencer. *The Probabilistic Method*. Wiley-Interscience, second edition, 2000.

2. Harry Buhrman, Tao Jiang, Ming Li, and Paul Vitányi. New applications of the incompressibility method: Part II. *Theoretical Computer Science*, 235(1):59–70, 2000.

3. Cristian Calude. *Information and Randomness: An Algorithmic Perspective*. Springer, 1994.

4. Gregory J. Chaitin. On the length of programs for computing finite binary sequences. *Journal of the ACM*, 13(4):547–569, 1966.

[6] If $\prec$ denotes the shortlex order on $\mathbf{B}^*$, then $x \prec y$ if either $l(x) < l(y)$ or $l(x) = l(y)$ and x is lexicographically smaller than y. This order is also called the length-plus-lexicographic order [15]

5. Gregory J. Chaitin. Information-theoretic limitations of formal systems. *Journal of the ACM*, 21(3):403–424, 1974.

6. Kurt Gödel. Über formal unentscheidbare Sätze der Principia mathematica und verwandter Systeme I. *Monatshefte für Mathematik und Physik*, 38:173–198, 1931.

7. A. N. Kolmogorov. On tables of random numbers. *Sankhyā: The Indian Journal of Statistics*, Series A, 70(4):369–376, 1963.

8. A. N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1(1):1–7, 1965.

9. Ming Li and Paul Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer, third edition, 2008.

10. Per Martin-Löf. The definition of random sequences. *Information and Control*, 9(6): 602–619, 1966.

11. Wolfgang J. Paul. Kolmogorov complexity and lower bounds. In Lothar Budach, editor, *Proceedings of the 1979 International Conference on Fundamentals of Computation Theory*, pages 325–334. Akademie-Verlag, 1979.

12. Panu Raatikainen. On interpreting Chaitin's incompleteness theorem. *Journal of Philosophic Logic*, 27(6):569–586, 1998.

13. Hartley Rogers, Jr. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, 1967. Reprinted, MIT Press, 1987.

14. Bertrand Russell. Mathematical logic as based on the theory of types. *American Journal of Mathematics*, 30(3):222–262, 1908.

15. Charles C. Sims. *Computation with Finitely Presented Groups*, volume 48 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 1994.

16. R. J. Solomonoff. A preliminary report on a general theory of inductive inference. Technical Report ZTB-138, Zator Company, November 1960.

17. Ray J. Solomonoff. A formal theory of inductive inference, part I. *Information and Control*, 7(1):1–22, 1964.

18. Ray J. Solomonoff. A formal theory of inductive inference, part II. *Information and Control*, 7(2):224–254, 1964.

19. A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, Series 2, 42:230–265, 1936.

20. Michiel van Lambalgen. Algorithmic information theory. *Journal of Symbolic Logic*, 54(4):1389–1400, 1989.