Institute for Language, Logic and Information

TWO DECADES OF APPLIED KOLMOGOROV COMPLEXITY

Ming Li
Paul M.B. Vitanyi

ITLI Prepublication Series
for Computation and Complexity Theory CT-88-01

University of Amsterdam

TWO DECADES OF APPLIED KOLMOGOROV COMPLEXITY

In Memoriam Andrei Nikolaevich Kolmogorov 1903-1987

Ming Li
Aiken Computation Laboratory, Harvard University

Paul M.B. Vitanyi
Centre for Mathematics and Computer Science
Department of Mathematics and Computer Science, University of Amsterdam

Received April 1988

--

Two Decades of Applied Kolmogorov Complexity
In Memoriam Andrei Nikolaevich Kolmogorov 1903-1987

Ming Li

Aiken Computation Laboratory, Harvard University, Cambridge, MA 02138

Paul M.B. Vitanyi

Centrum voor Wiskunde en Informatica and Universiteit van Amsterdam
Amsterdam, The Netherlands

ABSTRACT

This exposition is an introduction to the main ideas of Kolmogorov complexity and surveys the wealth of useful applications of this elegant notion. We distinguish: I) Application of the fact that some strings are compressible. This includes a strong version of Gödel's incompleteness theorem. II) Lower bound arguments that rest on application of the fact that certain strings cannot be compressed at all. Applications range from Turing machines to electronic chips. III) Probability Theory and a priori probability. Applications range from foundational issues to the theory of learning and inductive inference in Artificial Intelligence. IV) Resource-bounded Kolmogorov complexity. Applications range from NP-completeness and the P versus NP question to cryptography. (*Note*: A preliminary version of this paper appears in: *Proc. 3rd IEEE Structure in Complexity Theory Conference*, 1988. The final expanded version of this paper will appear in: *Handbook of Theoretical Computer Science* (J. van Leeuwen, Managing Editor), North-Holland.)

1. Introduction

The theory of computation is primarily concerned with the analysis and synthesis of algorithms in relation to the resources in time and space such algorithms require. R.J. Solomonoff, A.N. Kolmogorov and G.J. Chaitin (in chronological order) have invented an excellent theory of information content of strings, that is most useful for this pursuit. Intuitively, the amount of information in a finite string is the size (i.e., number of bits) of the smallest program that, started with a blank memory, computes the string and then terminates. A similar definition can be given for infinite strings, but in this case the program produces element after element forever. Thus, 1^n (a string of n 1's) contains little information because a program of size about $\log n$ outputs it. Likewise, the transcendental number $\pi=3.1415\cdots$, an infinite sequence of seemingly "random" decimal digits, contains $O(1)$ information. (There is a short program that produces the consecutive digits of π forever.) Such a definition would appear to make the amount of information in a string depend on the particular programming language used. Fortunately, it can be shown that all choices of programming languages (that make sense) lead to quantification of the amount of information that is invariant up to an additive constant.

The theory dealing with the quantity of information in individual objects goes by names such as "Algorithmic Information theory", "Kolmogorov complexity", "K-complexity", "Kolmogorov-Chaitin randomness", "Algorithmic complexity theory", "Descriptive complexity", "Program-Size complexity", and others. Although "Solomonoff-Kolmogorov-Chaitin complexity" would be the most appropriate name, we regard "Kolmogorov complexity" as well entrenched and commonly understood, and use it hereafter.

The mathematical theory of Kolmogorov complexity contains deep and sophisticated mathematics. Yet the amount of this mathematics one needs to know to apply the notions fruitfully in widely divergent areas, from recursive function theory to chip technology, is very little. However, formal knowledge does not necessarily imply the wherewithal to apply it, perhaps especially so in the case of Kolmogorov complexity. It is the

The work of the first author was supported in part by National Science Foundation Grant DCR-8606366, Office of Naval Research Grant N00014-85-k-0445, and Army Research Office Grant DAAL03-86-K-0171. His current address is: Department of Computer Science, York University, Ontario M3J 1P3, Canada (li@yuyetti.bitnet). Please send all correspondence to the second author at CWI, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands (paulv@piring.cwi.nl).

purpose of this survey to develop the minimum amount of theory needed, and briefly outline a scala of illustrative applications. In fact, while the pure theory of the subject will have its appeal to the select few, the surprisingly large field of its applications will, we hope, delight the multitude.

One can distinguish three application areas, according to the way Kolmogorov complexity is used. I.e., using the fact that some strings are extremely compressible; using the fact that many strings are not compressible at all; and using the fact that some strings may be compressed, but that it takes a lot of effort do so.

Kolmogorov complexity has its roots in probability theory, combinatorics, and philosophical notions of randomness, and came to fruition using the recent development of the theory of algorithms. Consider Shannon's classical information theory [119], that assigns a quantity of information to an ensemble of possible messages. All messages in the ensemble being equally probable, this quantity is the number of bits needed to count all possibilities. This expresses the fact that each message in the ensemble can be communicated using this number of bits. However, it does not say anything about the number of bits needed to convey any individual message in the ensemble. To illustrate this, consider the ensemble consisting of all binary strings of length 9999999999999999. By Shannon's measure, we require 9999999999999999 bits on the average to encode such a string. However, the string consisting of 9999999999999999 1's can be encoded in about 55 bits by expressing 9999999999999999 in binary and adding the repeated pattern 1. A requirement for this to work is that we have agreed on an algorithm that decodes the encoded string. We can compress the string still further when we note that 9999999999999999 equals $3^2 \times 1111111111111111$, and that 1111111111111111 consists of 2^4 1's.

Thus, we have discovered an interesting phenomenon: the description of some strings can be compressed considerably. In fact, there is no limit to the amount strings can be compressed, provided they exhibit enough regularity. This observation, of course, is the basis of all systems to express very large numbers, and was exploited early on by Archimedes in "The Sand Reckoner". However, if regularity is lacking, it becomes more cumbersome to express large numbers. For instance, it seems easier to compress the number ''one billion,'' than the number ''one billion seven hundred thirty five million two hundred sixty eight thousand and three hundred ninety-four,'' even though they are of the same order of magnitude.

This brings us to a related root of Kolmogorov complexity, the notion of randomness. In the context of the above discussion, random strings are strings that cannot be compressed. Now let us compare this with the common notions of mathematical randomness. To measure randomness, criteria have been developed which certify this quality. Yet, in recognition that they do not measure "true" randomness, we call these criteria "pseudo" random tests [57]. For instance, statistical survey of initial sequences of decimal digits of π have failed to disclose any significant deviations of randomness [57, 91]. But clearly, this sequence is so regular that it can be described by a simple program to compute it, and this program can be expressed in a few bits. Von Neumann [97]:

> ''Any one who considers arithmetical methods of producing random digits is, of course, in a state of sin. For, as has been pointed out several times, there is no such thing as a random number - there are only methods to produce random numbers, and a strict arithmetical procedure is of course not such a method. (It is true that a problem we suspect of being solvable by random methods may be solvable by some rigorously defined sequence, but this is a deeper mathematical question than we can go into now.)''

This fact prompts more sophisticated definitions of randomness. Notably R. von Mises [93] proposed notions that approach the very essence of true randomness. This is related with the construction of a formal mathematical theory of probability, to form a basis for real applications, in the early part of this century. This culminated in 1933 in A.N. Kolmogorov's classic treatment of the set theoretic axioms of the calculus of probability [60].

> ''This theory was so successful, that the problem of finding the basis of real applications of the results of the mathematical theory of probability became rather secondary to many investigators. ... [however] the basis for the applicability of the results of the mathematical theory of probability to real 'random phenomena' must depend in some form on the *frequency concept of probability*, the unavoidable nature of which has been established by von Mises in a spirited manner.'' [61].

The axioms of probability theory are designed so that abstract probabilities can be computed, but nothing is said about what probability really means, or how the concept can be applied meaningfully to the actual world. In [93] von Mises analyses the situation in detail, and suggests that a proper definition of probability depends on obtaining a proper definition of a random sequence.

The frequency theory to interpret probability says roughly, that if we perform an experiment many times, then the ratio of favorable outcomes to the total number n of experiments will, *with certainty*, tend to a limit, p say, as $n \rightarrow \infty$. This tells us something about the *meaning* of probability, namely the measure of the positive outcomes is p. But suppose we throw a coin 1000 times and wish to know what to expect. Is 1000 enough for convergence to happen? The statement above does not say. So we have to add something about the rate of convergence. But we cannot assert a *certainty* about a particular number of n throws, such as 'the proportion of heads will be $p \pm \varepsilon$ for large enough n (with ε depending on n)'. We can at best say 'the proportion will lie between $p \pm \varepsilon$ with at least such and such probability (depending on ε and n_0) whenever $n > n_0$'. But now we defined probability in an obviously circular fashion.

In 1919 von Mises proposed to eliminate the problem using the idea that there are random and non-random sequences. Roughly, if the random sequences form a set of full measure, and without exception to satisfy all laws of probability, then it seems then physically justifiable to assume that as a result of an (infinite) experiment only random sequences appear. We note that this does not represent the subtle position of von Mises himself. He postulates the existence of random sequences as certified by abundant empirical evidence, in the manner of physical laws like the Laws of Thermodynamics, and derives mathematical laws of probability as consequence. See [57, 82, 93, 143].

Von Mises' idea of an infinite random sequence of 0's and 1's (*Kollektiv*) is roughly as follows. Firstly, the limit of the frequency of 1's among the first n elements does exist. Secondly, for any "admissible" strategy of successively selecting infinitely many elements from the sequence the frequency of 1's in the selection goes to the same limit. The problem with this definition is that von Mises was unable to give a rigorous definition of what is the admissibility criterion. He essentially appeals to the familiar notion that no gambler, making a fixed number of wagers of "heads", at fixed odds and in fixed amounts, on the flips of a coin, can have profit in the long run from betting according to a system instead of betting at random. Says Church [33]: "this definition ... while clear as to general intent, is too inexact in form to serve satisfactorily as the basis of a mathematical theory."

It turns out that the obvious approach to a concrete mathematical formulation comes to grief as follows. An infinite sequence $a_1, a_2, ...$ of 0's and 1's is a random sequence if the following two conditions are satisfied. (1) If $f(r)$ is the number of 1's among the first r terms of the sequence, then $f(r)/r$ approaches a limit p, $0 < p < 1$, as r goes to infinity, and (2) for each partial function $\phi : \{0,1\}^* \rightarrow \{0,1\}$ satisfying: for all $r \in N$ there exist $n_r \in N$ such that

$$[\,|\{i : i \le n_r \text{ and } \phi(a(1)...a(i-1)) \in \{0,1\}\}| = r\,], \quad \text{(a)}$$

($|X|$ is the number of elements in set X) it holds that

$$\lim_{r\to\infty} \sum_{i=1}^{n_r} \frac{\phi(a(1)...a(i-1)) \cdot a(i)}{r} = p, \quad \text{(b)}$$

with p the same limit as in (1). Considering the sequence as fair coin tosses, condition (2) says there is no *strategy* ϕ (*Prinzip vom ausgeschlossenen Spielsystem*) that assures a player betting at fair odds to make infinite gain. However, since arbitrary functions are allowed as strategy, this definition is too restrictively, and no sequence exists that satisfies it with probability p other than 0.

Example. Let $a = a(1)a(2) \cdots$ be any infinite string satisfying (1). If a contains a finite number of 1's, then (2) implies $p=0$. Namely, by (a) ϕ has an infinite domain of definition, and therefore by (b) $p=0$. Assume now a contains an infinite number of 1's. Define ϕ_1 as $\phi_1(a(1)...a(i-1))=1$ if $a(i)=1$, and undefined otherwise. This satisfies (a), and by (b) $p=1$. However, this is not all of the story. Defining ϕ_0 by $\phi_0(a(1)...a(i-1))=b(i)$, $b(i)$ the complement of $a(i)$, for all i, we obtain by (b) that $p=0$. Consequently, if we allow functions like ϕ_1 and ϕ_0 as strategy, von Mises' definition can only be satisfied by strings with a finite number of 1's and $p=0$.

This counterexample was not recognized as such by von Mises, because it apparently violates the admissibility condition that $a(i)$ is not used in the definition of $\phi(a(1) \cdots a(i-1))$. However, in the example one does no such thing; it just happens that *after* a criterion for admissible ϕ is fixed, it turns out that for any sequence there is an admissible ϕ that coincides with a ψ that is defined in a clearly inadmissible fashion. We cannot here go into the various arguments put forward by the contestants in the ensuing discussion, but note that several attempts to resolve this problem turned out to be unsatisfactory one way or the other.

A. Wald proposed to restrict the admissible ϕ to a countable set [138]. This eliminated the above contradiction. A. Church proposed [33] to refine Wald's approach by restricting the notion of "strategy" ϕ from *any* partial function to the formal notion of any *effectively computable partial function*, as developed by A.M. Turing [127] and himself. He points out, that with computable ϕ, not only is the definition completely rigorous, and corresponding random sequences do exist, but moreover they are abundant since the infinite random sequences with $p=1/2$ form a set of measure one; and from the existence of random sequences with probability 1/2 the existence of random sequences associated with other probabilities is readily derived. Let us call sequences satisfying (1) and (2) with computable ϕ *Church-random*. Appeal to a theorem by Wald [138] yields as corollary that the set of Church-random sequences associated with any fixed probability has the cardinality of the continuum. Moreover, each Church-random sequence qualifies as a normal number. (A number is *normal* if each digit of the base, and each block of digits of any length, occurs with equal asymptotic frequency.) Note however, that not every normal number is Church-random. This follows, for instance, from Champernowne's number

$$0.1234567891011121314151617181920 \cdots$$

that is normal [30] and where the ith digit is easily calculated from i. The definition of a Church-random sequence implies that its consecutive digits can not be effectively computed. (Namely, existence of an effective ϕ_1 as above contradicts $0 < p < 1$ in (2).) Thus, an existence proof for Church-random sequences is necessarily non-constructive.

Unfortunately, the von Mises-Wald-Church definition is not yet good enough, since it was discovered by Ville [132] that even standard properties such as the law of iterated logarithm do not follow from it. In 1965, P. Martin-Löf, using ideas of Kolmogorov, succeeded in defining random sequences in a manner that is free of such difficulties [89]. His notion of infinite random sequences is related to infinite sequences of which all finite initial segments have high Kolmogorov complexity. For a survey of the work on infinite random sequences, see [65, 66].

Up till now the discussion has centered on infinite random sequences where the randomness is defined in terms of limits of relative frequencies. However,

> ''The frequency concept based on the notion of *limiting frequency* as the number of trials increases to infinity, does not contribute anything to substantiate the application of the results of probability theory to real practical problems where we always have to deal with a finite number of trials,'' [61].

It seems more appealing, to try to define randomness for finite strings first and only then define random infinite strings in terms of randomness of initial segments. The aim is to obtain a theory in which the existence of frequency limits follows from the randomness of the sequence, rather than the other way around [106]. However, properly defining random *finite* strings appeared to be an even more hopeless affair than such a definition for infinite strings. It was noted before, e.g. [61, 67], that ''randomness'' consists in lack of ''regularity'', and that, if some regularity can be caused by a simple law, then the chance that it is caused by this law is far greater than that it arose spontaneously. Moreover, it can be noted that there cannot be a very large number of simple laws. Identifying ''laws'' with ''algorithms'' brings us to our topic proper.

1.1. The Inventors

Kolmogorov complexity originated with the discovery of universal descriptions, and a recursively invariant approach to the concepts of complexity of description, randomness, and *a priori* probability. Historically, it is firmly rooted in R. von Mises' notion of random infinite sequences [93] as discussed above.

With the advent of electronic computers in the 1950's, a new emphasis on computer algorithms and a maturing general recursive function theory, ideas tantamount to Kolmogorov complexity came to many people's minds, because ''when the time is ripe for certain things, these things appear in different places in the manner of violets coming to light in early spring'' (Wolfgang Bolyai to his son Johann in urging him to claim the invention of non-Euclidean geometry without delay). Thus, with Kolmogorov complexity one can associate three inventors, in chronological order: R.J. Solomonoff in Cambridge, Massachusetts, [122], A.N. Kolmogorov in Moscow [61, 62], and G.J. Chaitin in New York [19, 20].

R.J. Solomonoff had been a student of R. Carnap at the University of Chicago in the fifties. His objective was to formulate a completely general theory of inductive inference that would overcome shortcomings of previous methods like [18]. To this purpose, in March 1964 he introduced what we term ''Kolmogorov'' complexity, a notion of *a priori* probability, and proved the Invariance Theorem (below), in a paper [122] that received little attention until Kolmogorov started to refer to it from 1968 onward. Already in November 1960 Solomonoff had published a *Zator Company* technical report on the subject [121]. It is interesting to note that these papers also contain informally the ideas of randomness of finite strings, noncomputability of Kolmogorov complexity, computability of approximations to the Kolmogorov complexity, and resource-bounded Kolmogorov complexity. A paragraph referring to Solomonoff's work occurs in [92]. To our knowledge, these are evidently the earliest documents outlining an algorithmic theory of descriptions.

In 1933 A.N. Kolmogorov* supplied probability theory with a powerful mathematical foundation [60]. Following a four decades long controversy on von Mises' concept of randomness, Kolmogorov finally introduced complexity of description of finite individual objects, as a measure of individual information content and randomness, and proved the Invariance Theorem in his paper of spring 1965 [62]. *Uspekhi Mat. Nauk* announced Kolmogorov's lectures on related subjects in 1961 and following years, and, says Kolmogorov: ''I came to similar conclusions [as Solomonoff], before becoming aware of Solomonov's work, in 1963-1964'' [63]. The new ideas were vigorously investigated by his associates. These included the Swedish mathematician P. Martin-Löf, visiting Kolmogorov in Moscow during 1964-1965, who investigated the complexity oscillations of infinite sequences and proposed a definition of infinite random sequences which is based on constructive measure theory. [89, 90]. L.A. Levin, then a student of Kolmogorov, defined *a priori* probability as a maximal semicomputable measure [143]. In 1974 he (and independently Chaitin in 1975) introduced Kolmogorov complexity based on self-delimiting programs [70]. This work relates to P. Gács' results concerning the differences between symmetric and asymmetric expressions for information [42].

G.J. Chaitin had finished the Bronx High School of Science, and was an 18 year old undergraduate student at the City College of the City University of New York, when he submitted [19, 20] for publication, in October and November 1965, respectively. Published in 1966, [19] investigated several ''state/symbol'' complexity variants of what we term ''Kolmogorov'' complexity, while [20], published in 1969, contained also the standard notion of Kolmogorov complexity, proves the Invariance Theorem, and studied infinite random binary sequences (in the sense of having maximally random finite initial segments) and their complexity oscillations. According to Chaitin: ''this definition [of Kolmogorov complexity] was independently proposed

* Andrei N. Kolmogorov, born 25 April 1903 in Tambov, USSR, died 20 October 1987 in Moscow. For biographical details see [3, 14, 41], and the obituary in the *Times* [98].

about 1965 by A.N. Kolmogorov and me ... Both Kolmogorov and I were then unaware of related proposals made in 1960 by Ray Solomonoff'' [22]. Chaitin in 1975 (and Levin independently in 1974) discovered and investigated Kolmogorov complexity based on self-delimiting programs [23]. For his recent work on Algorithmic Information theory (synonymous with Kolmogorov complexity theory), see e.g. [28,29].

Another variant of Kolmogorov complexity, viz., the length of the shortest program p that computes a function f such that $f(i)$ is the ith bit of the target string, was found by D.W. Loveland [84,85] and used extensively in [143]. Other variants and results were given by Willis [140], Levin [69], and Schnorr [116]. Apart from Martin-Löf's work, we mention that of C.P. Schnorr [114,115] on the relation between Kolmogorov complexity and randomness of infinite sequences.

1.2. Organization of the Paper

We treat the relevant mathematical notions of the theory of Kolmogorov complexity in the next Section. Of particular interest is application to probability theory and the notion of a priori probability. In the following Section, we apply the idea of compressibility of strings to the theory of algorithms and inductive inference, and in the Section after that we mention an application in Mathematics proper: prime number theorems. The bulk of the paper is contained in the Section that follows: applications of incompressibility to obtain lower bounds in a wide range of applications related to the theory of computing. In the penultimate Section we analyse various notions of resource-bounded Kolmogorov complexity, and its applications to structure in computational complexity.

2. Mathematical Theory

Kolmogorov gives a cursory but fundamental and elegant exposition of the basic ideas in [64]. The most complete treatment of the fundamental notions and results in Kolmogorov complexity is Zvonkin and Levin's 1970 survey [143]. Since this survey is not up to date, it should be complemented by Schnorr's [115,116] and Chaitin's more recent survey [25], or book [28]. For the advanced reader we mention Levin's [72]. There are several variants of Kolmogorov complexity; here we focus on the original version in [19,62,64,143]. (Later, we also define the more refined ''self-delimiting'' version.) First take a general viewpoint, as in [64], in which one assumes some domain D of objects with some standard enumeration of objects x by numbers $n(x)$. We are interested in the fact that $n(x)$ may not be the most economical way to specify x. To compare methods of specification, we agree to view such a method as a function S from natural numbers p written in binary notation to natural numbers n, $n=S(p)$. We do not yet assume that S is computable, but maintain full generality to show to what extent such a theory can also be developed with noneffective notions, and at which point effectiveness is required. For each object x in D we call the length $|p|$ of the smallest p that gives rise to it, the complexity of object x with respect to the specifying method S:

$$K_S(x) = \min\{|p| : S(p)=n(x)\},$$

and $K_S(x)=\infty$ if there are no such p. In computer science terminology we can call p a program and S a programming method (or language). Then one can say that $K_S(x)$ is the minimal length of a program to obtain x under programming method S. Considering distinct methods $S_1, S_2, \ldots, S_r$ of specifying the objects of D, it is easy to construct a new method S that gives for each object x in D a complexity $K_S(x)$ that exceeds only by c, c less than about $\log r$, the original minimum of the complexities $K_{S_1}(x), K_{S_2}(x), \ldots, K_{S_r}(x)$. The only thing we have to do is to reserve the first $\log r$ bits of p to identify the method S_i that should be followed, using as program the remaining bits of p. We say that a method S is ''stronger than a method S' with precision up to c'' if for all x:

$$K_S(x) \le K_{S'}(x) + c .$$

Above we have shown how to construct a method S that is stronger than any of the methods $S_1, \ldots, S_r$ with precision up to c, where $c \sim \log r$. Two methods S_1 and S_2 are called ''c-equivalent'', if each of them is c-stronger than the other. As Kolmogorov remarks, this construction would be fruitless if the hierarchy of methods with respect to strength were odd. However, under relatively natural restriction of S this is not so. Namely, among the computable functions S, *computable* in the sense of Turing [127], there exist *optimal* ones, that is, such that for any other computable function S':

$$K_S(x) \le K_{S'}(x) + c(S, S').$$

Clearly, all optimal methods of specifying objects in D are equivalent in the following way:

$$|K_{S_1}(x) - K_{S_2}(x)| \le c(S_1, S_2).$$

Thus, from an asymptotic point of view, the complexity $K(x)$ of an object x, when we restrict ourselves to optimal methods of specification, does not depend on accidental peculiarities of the chosen optimal method.

To fix thoughts, w.l.o.g. consider the problem of describing a finite string x of 0's and 1's. It is useful to develop the idea that the complexity of specifying an object can be facilitated when another object is already specified. Thus, we define the complexity of an object x, given an object y. Let $d, y, x \in \{0,1\}^*$. Any computable function f together with strings d, y, such that $f(d,y)=x$, is such a description. The descriptional, or *Kolmogorov*, complexity K_f of x, *relative* to f and y, is defined by

$$K_f(x|y) = \min\{|d| : d \in \{0,1\}^* \ \& \ f(d,y)=x\},$$

and $K_f(x|y)=\infty$ if there are no such d. Here $|d|$ denotes the *length* (number of 0's and 1's) of d.

Invariance Theorem (Solomonoff [122], Kolmogorov [62], Chaitin [20]). *There exists a partial recursive function f_0, such that, for any other partial recursive function f, there is a constant c_f such that for* *all strings* x, y, $K_{f_0}(x \mid y) \le K_f(x \mid y) + c_f$.

Proof. Fix some standard enumeration of Turing machines, and let $n(T)$ be the number associated with Turing machine T. Let f_0 be the *universal* partial recursive function computed by a universal Turing machine U, such that U started on input $0^n 1p$, $p \in \{0,1\}^*$, simulates T on input p, for $n(T)=n$. For convenience in the proof, we choose U such that if T halts, then U first erases everything apart from the halting contents of T's tape, and also halts. By construction, for each $p \in \{0,1\}^*$, T started on p eventually halts iff U started on $0^{n(T)}1p$ eventually halts. Choosing $c_f = n(T)+1$ finishes the proof. □

Clearly, a function f_0 that satisfies the Invariance Theorem is *optimal* in the sense discussed above. Therefore, we set the canonical *conditional Kolmogorov* complexity $K(x \mid y)$ of x under condition of y equal to $K_{f_0}(x \mid y)$, for some fixed optimal f_0. Define the *unconditional Kolmogorov* complexity of x as $K(x) = K(x \mid \varepsilon)$, where ε denotes the empty string ($|\varepsilon| = 0$). Before we continue, we recall the definitions of the big-O notation.

Notation (order of magnitude). We use the *order of magnitude* symbols O, o, Ω and Θ. If f and g are functions on the real numbers, then (i) $f(x) = O(g(x))$ if there are positive constants c, x_0, such that $f(x) \le cg(x)$, for all $x \ge x_0$; (ii) $f(x) = o(g(x))$, if $\lim_{x \to \infty} f(x)/g(x) = 0$; (iii) $f(x) = \Omega(g(x))$ if $f(x) \ne o(g(x))$; (iv) and $f(x) = \Theta(g(x))$ if both $f(x) = O(g(x))$ and $f(x) = \Omega(g(x))$. The relevant properties are extensively discussed in [58, 133]. This use of Ω was introduced first by Hardy and Littlewood in 1914, and must not be confused by Chaitin's real number Ω we meet in a later section.

Example. For each finite binary string x we have $K(xx) \le K(x) + O(1)$. Namely, let T compute x from program p. Now fix a universal machine V which, on input $0^{n(T)}1p$, simulates T just like the reference machine U in the proof of the Invariance Theorem, but additionally V doubles T's output before halting. Now V started on $0^{n(T)}1p$ computes xx, and therefore U started on $0^{n(V)}10^{n(T)}1p$ computes xx. Hence, for all x, $K(xx) \le K(x) + n(V) + 1$.

Example. It is seductive to conjecture $K(xy) \le K(x) + K(y) + O(1)$, the obvious (but false) argument running as follows. Suppose we have a shortest program p to produce x, and a shortest program q to produce y. Then with $O(1)$ extra bits to account for some Turing machine T that schedules the two programs, we have a program to produce x followed by y. However, any such T will have to know where to divide its input to identify p and q. We can separate p and q by prefixing pq by a clearly distinguishable encoding r of the length $|p|$ in $O(\log |p|)$ bits (see next Section on self-delimiting strings). Consequently, we have at best established $K(xy) \le K(x) + K(y) + O(\log(\min(K(x), K(y))))$.

2.1. Incompressibility

Since there are 2^n binary strings of length n, but only $2^n - 1$ possible shorter descriptions d, it follows that, for all n, there is a binary string x of length n such that $K(x) \ge n$. We call such strings *incompressible*. It also follows that, for any length n and any binary string y, there is a binary string x of length n such that $K(x \mid y) \ge n$.

Example. Are all substrings of finite incompressible strings also incompressible? A string $x = uvw$ can be specified by a description of v, literal descriptions of the binary representation of $|u|$ and the concatenation uw. Additionally, we need information how to tell these three items apart. Such information can be provided in $O(\log |x|)$ bits. Thus,

$$K(x) \le K(v) + O(\log |x|) + |uw| \ ,$$

(All logarithms in this paper are base 2, unless it is explicitly noted they are not.) Hence, for random strings x with $K(x) \ge |x|$ we obtain

$$K(v) \ge |v| - O(\log |x|) \ .$$

Example. Define $p(x)$ as a shortest program for x. We show that $p(x)$ is incompressible. There is a constant $c > 0$, such that for all strings x, we have $K(p(x)) \ge c \cdot |p(x)|$. For suppose the contrary, and there is a program $p(p(x))$ that generates $p(x)$ with $|p(p(x))| \le c \cdot |p(x)|$. Define a universal machine V that works just like the reference machine U, except that V first simulates U on its input to obtain an output, and then uses this output as input on which to simulate U once more. But then, U with input $0^{n(V)}1p(p(x))$ computes x, and therefore $K(x) \le c \cdot |p(x)| + n(V) + 1$. However, this yields $(1-c)K(x) \le n(V) + 1$, for all x, which is impossible by a trivial counting argument. Similarly we can show that there is a $c > 0$ such that for all strings x, we have $K(p(x)) \ge |p(x)| - c$.

Example. It is easy to see that $K(x \mid x) \le n(T) + 1$, where T is a machine that just copies the input to the output. However, it is more interesting that $K(p(x) \mid x) \le \log K(x) + O(1)$, which cannot be improved in general. Hint: later we show that K is a noncomputable function. This rules out that we can compute $p(x)$ from x. However, we can dovetail the computation of all programs shorter than $|x|+1$: run the 1st program 1 step, run the 1st program 1 step and 2nd program 1 step, and so on. This way we will eventually enumerate all programs that output x. However, since some computations may not halt, and the halting problem is undecidable, we need to know the length of a shortest program $p(x)$ to recognize such a program when it is found.

A natural question to ask is: how many strings are incompressible? It turns out that virtually all strings of given length n are incompressible. Namely, let $g(n)$ be an unbounded integer function. Call a string x of length n, g-*incompressible* if $K(x) \ge n - g(n)$. There are 2^n binary strings of length n, and only $2^{n-g(n)} - 1$ possible descriptions shorter than $n - g(n)$. Then, considering the set of strings of length n, the ratio between the number of strings x with $K(x)<n-g(n)$ and the total number of strings is at most $2^{-g(n)+1}$. Thus, in a natural sense, the g-incompressible finite strings have measure one in the set of all finite strings.

If we want to consider infinite strings, incompressibility can be defined as follows. An infinite string x is *g-incompressible* if each initial string x_n of length n has $K(x_n)\geq n-g(n)$, from some n onward. Choose for g, say $g(n)=2\log n$, and define *incompressibility* as $(2\log n)$-incompressibility. In [89], Martin-Löf has defined a satisfactory notion for randomness of infinite strings. Martin-Löf random strings are $(2\log n)$-incompressible, but not $(\log n)$-incompressible, cf. later. Chaitin's [20] contains a related result. We call finite or infinite incompressible strings loosely 'random' or 'Kolmogorov random', but want to stress here that randomness for infinite strings according to Martin-Löf has a more profound definition. We return in somewhat more detail to this matter below.

Curiously, though most strings are random, it is impossible to effectively prove them random. The fact that almost all finite strings are random but cannot be proved to be random amounts to Chaitin's information-theoretic version of Gödel's theorem below. Strings that are not incompressible are *compressible* or *nonrandom.* In a natural sense, they have measure zero in the set of strings (both finite and infinite).

We have now formalized the essence of what we need for the applications in the sequel. Having made our notions precise, many applications can be described informally yet rigorously. Apart from the next Section on how to construct efficient descriptions, the remainder of the theory of Kolmogorov complexity we treat below is not in general required for the later applications.

2.2. Self-delimiting Descriptions

In previous Sections we formalized the concept of a greatest lower bound on the length of a description. Now we look at feasibility. Let the variables x, y, x_i, y_i ... denote strings in $\{0,1\}^*$. A *description* of x, $|x|=n$, can be given as follows.

(1) A piece of text containing several formal parameters $p_1,\ldots,p_m$. Think of this piece of text as a formal parametrized procedure in an algorithmic language like PASCAL. It is followed by

(2) an ordered list of the actual values of the parameters.

The piece of text of (1) can be thought of as being encoded over a given finite alphabet, each symbol of which is coded in bits. Therefore, the encoding of (1) as prefix of the binary description of x requires $O(1)$ bits. This prefix is followed by the ordered list (2) of the actual values of $p_1,\ldots,p_m$ in binary. To distinguish one from the other, we encode (1) and the different items in (2) as self-delimiting strings, an idea used already by Chaitin in [19,20], as follows.

For natural numbers n, let $\text{bin}(n)\in\{0,1\}^*$ be the binary representation of n without leading zeros. For each string $w\in\{0,1\}^*$, the string $\bar{w}$ is obtained by inserting a "0" in between each pair of adjacent letters in w, and adding a "1" at the end. I.e.,

$$\overline{01011}=0010001011\ .$$

Let $w'=\overline{\text{bin}(|w|)}w$. The string w' is called the *self-delimiting* version of w. So '10001101011' is the self-delimiting version of '01011'. The self-delimiting binary version of a positive integer n requires $\log n+2\log\log n$ bits, and the self-delimiting version of a binary string w requires $|w|+2\log|w|$ bits. For convenience, we denote the length $|\text{bin}(n)|$ of a natural number n by "$\log n$".

Example (generalization). More generally, for $w\in\{0,1\}^*$, $d_0(w)=\bar{w}$ is the self-delimiting version of order 0 of w using $2|w|$ bits. Above we defined the "standard" self-delimiting version $d_1(w)=w'$ of order 1. In general, for $i\geq 1$, $d_i(w)=\bar{w}_i w_{i-1}\ldots w_1 w$, with $w_1=\text{bin}(|w|)$ and $w_j=\text{bin}(|w_{j-1}|)$ $(1<j\leq i)$, is the self-delimiting version of order i of w. Define $\log^{(1)}=\log$, and $\log^{(j+1)}=\log\log^{(j)}$ for $j\geq 1$. Then,

$$|d_i(w)|=|w|+\log^{(1)}|w|+\cdots+\log^{(i-1)}|w|+2\log^{(i)}|w|.$$

Obviously, further improvements are possible.

Example. Self-delimiting descriptions were used in the proof of the Invariance Theorem. (Namely, in the encoding $0^{n(T)}1$.) Using it explicitly, we can define Kolmogorov complexity as follows. Fix an effective coding C of all Turing machines as binary strings such that no code is a prefix of any other code. Denote the code of Turing machine M by $C(M)$. Then the Kolmogorov complexity of $x\in\{0,1\}^*$, with respect to C, is defined by $K_C(x)=\min\{|C(M)y| : M \text{ on input } y \text{ halts with output } x\}$.

Example (self-delimiting Kolmogorov complexity). A code C such that $C(x)$ is not a prefix of $C(y)$ if $x\neq y$ is called a *prefix code*. We can also define Kolmogorov complexity by requiring at the outset that the set of descriptions form an effective prefix code. The resulting variant, called *self-delimiting* Kolmogorov complexity, has nicer mathematical properties than the original one, and has therefore become something of the standard one in the field. This complexity is variously denoted in the literature by KP, I, or simply by K if no confusion with the original can result. We treat it in a later section and denote it there by J.

Example (partially obscure string). In proving lower bounds in the Theory of Computation it is sometimes useful to give an efficient description of an incompressible string with 'holes' in it. The reconstruction of the complete string is then achieved using an additional description. In such an application we aim for a contradiction where these two descriptions together have significantly smaller length than the incompressible string they describe. Formally, let $x_1\cdots x_k$ be a binary string of length n with the x_i's $(1\leq i\leq k)$ blocks of equal length C. Suppose that d of these blocks are deleted and the relative distances in between deleted blocks are known. We can describe this information by: (1) a formalization of this discussion in $O(1)$ bits, and

(2) the actual values of

$$C, m, p_1, d_1, p_2, d_2, \ldots, p_m, d_m,$$

where m ($m \leq d$) is the number of "holes" in the string, and the literal representation of

$$\hat{x} = \hat{x}_1 \hat{x}_2 \cdots \hat{x}_k .$$

Here $\hat{x}_i$ is x_i if it is not deleted, and is the empty string otherwise; p_j, d_j indicates that the next p_j consecutive x_i's (of length C each) are one contiguous group followed by a gap of $d_j C$ bits long. Therefore, $k-d$ is the number of (non-empty) $\hat{x}_i$'s, with

$$k = \sum_{i=1}^{m} p_i + d_i \quad \& \quad d = \sum_{i=1}^{m} d_i \ .$$

The actual values of the parameters and $\hat{x}$ are coded in a self-delimiting manner. Then, by the convexity of the logarithm function, the total number of bits needed to describe the above information is no more than (loglog≤log):

$$\textstyle\sum_{i=1}^{k} |\hat{x}_i| + 3d \log(k/d) + O(\log n).$$

2.3. Quantitative Estimate of K

We want to get some insight in the quantitative behavior of K. We follow [143]. We start this Section with a useful property. Consider the conditional complexity of a string x, with x an element of a given finite set M, given some string y. Let $||M||$ denote the number of elements in M. Then the fraction of $x \in M$ for which $K(x|y) \leq |\text{bin}(||M||)| - m$, does not exceed 2^{-m+1}. Namely, if $K(x|y) < n$ then there is a program p of length at most n such that $f_0(p,y) = x$. Hence there cannot be more such x than there are such p, which is at most $2^{n+1}-1$. Moreover, $||M|| \geq 2^{|\text{bin}(||M||)|}-1$. Combining these two observations, we find $(2^{|\text{bin}(||M||)|-m+1}-1)\ (2^{|\text{bin}(||M||)|}-1)^{-1} < 2^{-m+1}$. Hence we have shown that the conditional complexity of the majority of elements in a finite set cannot be significantly less than the complexity of the size of that set. The following Lemma says that it can not be significantly more either.

Lemma. *Let A be an r.e. set of pairs (x,y), and let $M_y = \{x : (x,y) \in A\}$. Then, up to a constant depending only on A, $K(x|y) \leq |\text{bin}(||M_y||)|$.*

Proof. Let A be enumerated by a Turing machine T. Using y, modify T to T_y such that T_y enumerates the first all pairs (x,y) in A, without repetition. In order of enumeration we select the p'th pair (x,y), and output the first element, i.e. x. Then we find $p < ||M_y||$, such that $T_y(p) = x$. Therefore, we have by the Invariance Theorem $K(x|y) \leq K_{T_y}(x) \leq |\text{bin}(||M_y||)|$, as required. □

Example. Let A be a subset of $\{0,1\}^*$. Let $A^{\leq n}$ equal $\{w \in A : |w| \leq n\}$. If $||A^{\leq n}|| / (2^{n+1}-1)$ is $O(g(n))$, then we call A g-sparse. For example, the set of all finite strings that have twice as many 0's as 1's is n^{-1}-sparse. This has as consequence that all but finitely many of these strings have short programs.

Claim. (a) If A is recursive and g-sparse, g is $o(1)$, then for all constant c there are only finitely many x in A with $K(x) \geq |x| - c$.
(b) If A is r.e. and $O(n^{-(1+\varepsilon)})$-sparse, $\varepsilon > 0$, then, for all constant c, there are only finitely many x in A with $K(x) \geq |x| - c$.
(c) If A is r.e. and $|A^{\leq n}| \leq p(n)$ with p a polynomial, then, for all constant $c > 0$, there are only finitely many x in A with $K(x) \geq |x|/c$.

Proof. (a) Consider the lexicographic enumeration of all elements of A. There is a constant d, such that the ith element x of A has $K(x) \leq K(i) + d$. If x has length n, then the sparseness of A implies that $K(i) \leq n - \log g(n) + O(1)$. Therefore, for each constant c, and all n, if x in A is of length n then $K(x) < n - c$, from some n onward.
(b) Fix c. Consider an enumeration of n-length elements of A. For all such x, the Lemma above in combination with the sparseness of A implies that $K(x|n) \leq n - (1+\varepsilon)\log n + O(1)$. Therefore, $K(x) \leq n - \varepsilon \log n + O(1)$, and the right hand side of the inequality is less than $n-c$ from some n onward.
(c) Similarly to above. □

We now look at unconditional complexity.

Lemma. *For any binary string x:*
(a) $K(x) \leq |x|$, up to some constant not depending on x.
(b) The fraction of x for which $K(x) < l - m$ and $|x| = l$ does not exceed 2^{-m+1}, so that equality holds in (a) for the majority of words.
(c) $\lim_{x \to \infty} K(x) = \infty$, and
(d) for $m(x)$ is the largest monotonic increasing integer function bounding $K(x)$ from below: $m(x) = \min_{y \geq x} K(y)$, we have $\lim_{x \to \infty} m(x) = \infty$.
(e) For any partial recursive function $\phi(x)$ tending monotonically to ∞ from some x_0 onwards, we have $m(x) < \phi(x)$.
(f) $|K(x+h) - K(x)| \leq 2|h|$, up to some constant independent of x,h. I.e., although $K(x)$ varies all the time between $|x|$ and $m(x)$, it does so fairly smoothly.

Proof. (a)-(d) have been argued above or are easy. For (e) see [143]. We prove (f). Let p be a minimal length description of x so that $K(x) = |p|$. Then we can describe $x+h$ by $\bar{h}p$, $\bar{h}$ the (order 0) self-delimiting description of h, and a description of this discussion in a constant number of bits. Since $|\bar{h}| \leq 2|h|$ (see previous Section), this proves (f). □

Example (non-monotonicity). (Inequalities below are up to an independent constant.) Let $ww \in \{0,1\}^*$, $|w| = n$. Then $K(ww) = K(w) + O(1)$, but dividing $w = uv$ in a sufficiently random way gives $K(wu) \geq K(w) + \varepsilon \log n$, for some $\varepsilon > 0$.

Example (non-monotonicity). Let $w \in \{0,1\}^*$, $w = xy$, $|w| = n$, such that $y = \text{bin}(n)$. I.e., the last $\log n$ bits of w consist of the binary encoding of the length of w. Then $K(w) = K(x) + O(1)$ ($\leq n - \log n + O(1)$). I.e., w can be described by its suffix property and x. By dividing $y = uv$ in a sufficiently random way, it is easy to see that $K(xu) \geq K(x) + \varepsilon \log\log n$ for some $\varepsilon > 0$. (Note that

$K(w)=K(w^R)$ where w^R is w written backwards.)

Example. For any binary string w, $|w|=n$, we have $K(w|n)\leq K(w)$, but usually the length of w does not give too much information about w. But sometimes it does. For instance, consider the self-delimiting description w', $|w'|=m$, of a string $w=0^n$. Then, $K(w')=K(n)+O(1)$, but $K(w'|n)=O(1)$.

These examples show that K is *non-monotonic*, i.e., if x is a prefix of y then not necessarily $K(x)\leq K(y)$. One reason to introduce the self-delimiting version of Kolmogorov complexity (below) is to have monotonicity. In the next Section we look at the behavior of K on prefixes of infinite strings.

2.4. Infinite Random Strings

In connection with what has been said before, we would like to call an infinite binary string x random if there is a constant C such that, for all n, the n-length prefix x_n has $K(x_n)\geq n-C$. However, such strings do not exist [20, 89, 90]:

Theorem (Chaitin, Martin-Löf). *If $f(n)$ is a computable function such that $\sum 2^{-f(n)}=\infty$, then for any infinite binary sequence x there are infinitely many n for which $K(x_n)<n-f(n)$. (Here x_n is the n-length prefix of x.)*

Example. $f(n)=\log n$ satisfies the condition of the theorem. Let $x=x(1)x(2)\cdots$ be any infinite binary string, and x_m any m-length prefix of x. Assume $x(1)=1$. If n is a natural number such that $\mathrm{bin}(n)=x_m$, $m=\log n$, then $K(x_n)=K(x(m+1)\cdots x(n))+O(1)$. Namely,

$$2^{m-1}\leq n=\sum_{i=0}^{m-1}x(m-i)2^i\leq 2^m.$$

Therefore, with $O(1)$ additional bits of information, we can easily reconstruct $\mathrm{bin}(n)$ from length $n-m$ of $x(m+1)\cdots x(n)$.

However, it was observed by Chaitin and Martin-Löf that if $f(n)$ is such that the series

$$\sum 2^{-f(n)} \tag{1}$$

converges constructively (for example $f(n)=\log n+2\log\log n$), then almost all strings x (in the the sense of binary measure) have the property

$$K(x_n)\geq n-f(n), \tag{2}$$

from some n onwards. For details see [20, 90, 143]. Due to these complexity oscillations the idea of identifying infinite random sequences with those such that $K(x_n)\geq n-C$ does not work. Martin-Löf observed that to justify any proposed definition of randomness one has to show that the sequences, which are random in the stated sense, satisfy the several properties of stochasticity we know from the theory of probability. Instead of proving each such property separately, one may be able to show, once and for all, that the random sequences introduced possess, in an appropriate sense, all possible properties of stochasticity.

Using constructive measure theory, Martin-Löf [89] develops the notion of random binary sequences as having all 'effectively verifiable' properties that from the point of view of the usual probability theory are satisfied with 'probability 1'. I.e., they pass all effective statistical tests for randomness in the form of a 'universal' test, where the bits represent the outcome of independent experiments with outcomes 0 or 1 with probability 1/2. Using this definition of randomness he shows [90]:

Theorem (Martin-Löf). *Random binary sequences satisfy (2) from some n onwards, provided (1) converges constructively.*

Theorem (Martin-Löf). *If $K(x_n|n)\geq n-C$ for some constant C and infinitely many n, then x is a random binary sequence.*

For related work see also [84, 114, 115, 117, 143].

2.5. Algorithmic Properties of K

We select some results from Zvonkin and Levin's survey [143]. One can consider $K(x)$ as a function that maps a positive integer $1x$ to a positive integer $K(x)$.

Theorem. *(a) The function $K(x)$ is not partial recursive. Moreover, no partial recursive function $\phi(x)$, defined on an infinite set of points, can coincide with $K(x)$ over the whole of its domain of definition.*
(b) There is a (total) recursive function $H(t,x)$, monotonically decreasing in t, such that $\lim_{t\to\infty}H(t,x)=K(x)$. I.e., we can obtain arbitrary good estimates for $K(x)$ (but not uniformly).

Proof. We prove (a). For the proof of (b) see [143]. Every infinite r.e. set contains an infinite recursive subset, Theorem 5-IV in [111]. Select an infinite recursive set V in the domain of definition of $\phi(x)$. The function $F(m)=\min\{x: K(x)\geq m, x\in V\}$ is (total) recursive (since $K(x)=\phi(x)$ on V), and takes arbitrary large values. Also, by construction $K(F(m))\geq m$. On the other hand, $K(F(m))\leq K_F(F(m))+c_F$ by definition of K, and obviously $K_F(F(m))\leq|\mathrm{bin}(m)|$. Hence, $m\leq\log m$ up to a constant independent of m, which is false. □

Remark. This is related to Chaitin's version of Gödel's incompleteness result, cf. below.

It turns out that with Kolmogorov complexity one can quantify the distinction between r.e. sets and recursive sets. Let $x=x(1)x(2)\cdots$ be an infinite binary sequence such that the set of numbers n with $x(n)=1$ is r.e. If the complementary set with the $x(n)=0$ were also r.e., then $f(n)=x(n)$ would be computable, and the relative complexity $K(x_n|n)$ bounded (x_n is the n-length prefix of x). But in the general case, when the set of 1's is r.e., $K(x_n|n)$ can grow unboundedly.

Theorem (Barzdin', Loveland). *For any binary sequence x with the set $M=\{n: x(n)=1\}$ is r.e. holds $K(x_n|n)\leq\log n+c_M$, where c_M is a constant dependent on M (but not dependent on n). Moreover, there are such sequences such that for any n holds $K(x_n|n)\geq\log n$.*

For a proof see [7, 84, 143]. In [64] Kolmogorov gives the following interesting interpretation with respect to investigations in the foundations of mathematics. Viz., label all Diophantine equations by natural numbers. Matiyasevich has proved that there is no general algorithm to answer the question whether the equation D_n is soluble in integers (the answer to Hilbert's 10th problem is negative). Suppose, we weaken the problem by asking for the existence of an algorithm that enables us to answer the question of the existence or nonexistence of solutions for the first n Diophantine equations with the help of some supplementary information of size related to n. The theorem above shows that this size can be as small as $\log n + C$. Such information is in fact contained in the $\sim\log n$ length prefix of the mythical number Ω, that encodes the solution to the halting problem for the first n Turing machines, cf. later.

In the same 1968 paper [7] Barzdin' derives one of the first results in 'time-limited' Kolmogorov complexity. It shows that by imposing recursive time limits on the decoding procedure, the length of the shortest description of a string can sharply increase. Let t be an integer function. Define $K^t(x_n \mid n)$ as the minimum length of a program p such that the universal machine U started on $n'p$ computes the n-length prefix of x within $t(n)$ steps, and then halts. (n' is the self-delimiting description of n.)

Theorem (Barzdin'). *For any binary sequence x with an r.e. set $M=\{n : x(n)=1\}$ and any constant $c>0$, there exists a (total) recursive function t such that for infinitely many n holds $K^t(x_n \mid n) \le cn$. Moreover, there are such sequences such that for any (total) recursive t and any n holds $K^t(x_n \mid n) \ge c_t n$, with c_t is a constant independent of n (but dependent on t.)*

2.6. Information

If the conditional complexity $K(x \mid y)$ is much less than the unconditional complexity $K(x)$, then we may interpret this as an indication that y contains much information about x. Consequently, up to an additive constant, we can regard the difference

$$I(x:y) = K(x) - K(x \mid y)$$

as a quantitative measure of the information about x contained in y. If we choose f_0 such that $f_0(\varepsilon, x) = x$, then

$$K(x \mid x) = 0, \quad I(x:x) = K(x).$$

In this way we can view the complexity $K(x)$ as the information contained in an object about itself. For applications, this definition of the quantity of information has the advantage that it refers to individual objects, and not to objects treated as elements of a set of objects with a probability distribution given on it, as in [119]. Does the new definition have the desirable properties that hold for the analogous quantities in classic information theory? We know that equality and inequality can hold only up to additive constants, according to the indeterminacy in the Invariance Theorem. For example, the equality $I(x:y) = I(y:x)$ cannot be expected to hold exactly, but a priori it can be expected to hold up to a constant related to the choice of reference function f_0. However, with the current definitions, information turns out to be symmetric only up to a logarithmic factor. Define $K(x,y)$ as the complexity of x and y together. I.e., the length of the least program of U that prints out x and y and a way to tell them apart. The following Lemma is due to Kolmogorov and Levin.

Lemma (Symmetry). *To within an additive term of $O(\log K(x,y))$,*

$$K(x,y) = K(x) + K(y \mid x)$$

It then follows immediately that, to within an additive term of $O(\log K(x,y))$,

$$K(x) - K(x \mid y) = K(y) - K(y \mid x),$$

and therefore:

$$|I(x:y) - I(y:x)| = O(\log K(x,y)).$$

It has been established that the difference can be of this order. See [143].

2.7. Self-Delimiting Kolmogorov Complexity

In 1974 Levin and Gács [42, 70], and in 1975 Chaitin [23], discovered and analysed the notion of self-delimiting Kolmogorov complexity J. This more refined version is, in a sense, implicit in Solomonoff's original a priori probability [121, 122]. For the development of the theory this is a more satisfactory complexity measure than the K-version, but for the applications below one can generally use both equally well. Differences between J and K are that J is monotonic within an additive constant (rather than the logarithm) and extendible to the case of infinite sequences without the logarithmic fudge term.

Note. With some abuse of notation, everywhere else than in this Section we denote all types of Kolmogorov complexity simply by K. Whether we mean the non-self-delimiting one or the self-delimiting one, will be clear from the context in case it matters, but most often the application is too crude to discriminate.

Consider a class of Turing machines with a one-way input tape, a one-way output tape, and a two-way work tape. Let the infinite input tape contain only 0's or 1's (no blanks). Let the symbols on the input tape be provided by independent tosses of an unbiased coin. For a machine M and each binary string s, define $P(s)$ as the *probability* that M eventually halts with s written on the output tape. (Solomonoff has called P the *a priori* probability, cf. later.) The *entropy* $H(s) = -\log P(s)$. We call a binary string p a *program* for M if M starts scanning the leftmost bit of p and halts scanning the rightmost bit of p. The *information* $J(s)$ is the length of the shortest program p that outputs s.

Fact. We have defined programs so that no program is the prefix of another one. Each program is *self-delimiting* with respect to M.

This enables us to give a natural probability distribution over programs: the probability of program p is simply $2^{-|p|}$. We now can easily compose programs from self-delimiting subprograms by prefixing a sequence of n self-delimiting programs with a self-delimiting description of n. Choosing the method in the previous Section, we can encode a binary string s by a program of length $|s|+2\log|s|$. Namely, define Turing machine M such that it outputs a binary string s iff it first reads the self-delimiting binary encoding of the length of s, and then the usual binary representation of s. Thus, with respect to M, $P(s) \geq 2^{-|s|-2\log|s|}$, $H(s) \leq |s|+2\log|s|$, and $J(s) \leq |s|+2\log|s|$.

To make the definitions meaningful, we normalize these measures with respect to an optimal *universal* machine chosen such that it maximizes P and minimizes H and J. Let U be such a machine. The relation between the different notions is $K(s) \leq J(s) \leq H(s)$.

Example. To within an additive constant, for all finite binary strings x,y we have $J(xy) \leq J(x)+J(y)$. Namely, let p and q be self-delimited programs for x and y, respectively. Let V be a universal machine just like the reference machine U, except that it simulates U first on p to produce x, then on q to produce y, and subsequently outputs xy. Presented with input pq, V can tell p apart from q because p is self-delimited. Hence, U with program $0^{n(V)}1pq$, computes xy. Therefore, for all finite binary strings x,y, $J(xy) \leq J(x)+J(y)+n(V)+1$.

The self-delimiting complexity J satisfies many laws without a logarithmic fudge term, e.g. *monotonicity*: if x is a prefix of y then, within an additive constant independent of x,y we have $J(x) \leq J(y)$. Infinite random sequences can be naturally defined using J complexity. The following lemma, due to Chaitin, Gács, Levin, shows that $J(x)$ is a symmetric measure of the information in x:

Lemma (Strong Symmetry). *To within an additive constant,*

$$J(x,J(x)) = J(x).$$

$$J(x,y) = J(x) + J(y \mid (x,J(x)))$$

2.8. Probability Theory

P.S. Laplace [67] has pointed out the following conflict between our intuition and the classical theory of probability:

> "In the game of heads and tails, if head comes up a hundred times in a row then this appears to us extraordinary, because the nearly infinite number of combinations that can arise in a hundred throws are divided in regular sequences, or those in which we observe a rule that is easy to grasp, and in irregular sequences, that are incomparably more numerous."

Yet, 100 heads are just as probable as any other sequence of heads and tails, even though we feel that it is less 'random' than some others. We can formalize this (following P. Gács [43]). Let us call a *payoff* function with respect to distribution P any nonnegative function $t(x)$ with $\sum_x P(x)t(x) \leq 1$. Suppose our favorite nonprofit casino asks 1 dollar for a game that consists of a sequence of flips of a fair coin, and claims that each outcome x has probability $P(x)=2^{-|x|}$. To back up this claim, it must agree to pay $t(x)$ dollars on outcome x. Accordingly, we propose a payoff t_0 with respect to P_n (P restricted to sequences of n coin flips): put $t_0=2^{n/2}$ for all x whose even digits are 0 (head), and 0 otherwise. This bet will cost the casino $2^{50}-1$ dollars for the outcome ($n=100$) above. Since we must propose the payoff function beforehand, it is unlikely that we define precisely the one that detects this particular fraud. However, fraud implies regularity, and, as Laplace suggests, the number of regular bets is so small that we can afford to make all of them in advance.

> "If we seek a cause wherever we perceive symmetry, it is not that we regard the symmetrical event as less possible than the others, but, since this event ought to be the effect of a regular cause or that of chance, the first of these suppositions is more probable than the second."

Let us make this formal, using some original ideas of Solomonoff as developed and made precise by Levin and Gács. For most binary strings of length n, no significantly shorter description exists, since the number of short descriptions is small. We can sharpen this observation by, instead of counting the number of simple sequences, measuring their probability. Consider only descriptions $\bar{x}$ of x, that are first among the shortest self-delimiting descriptions of x. The correspondence $x \rightarrow \bar{x}$ is a *code* in which no codeword is a prefix of another one. Then, it can be shown that

$$\sum_x 2^{-K(x \mid y)} \leq 1\,, \tag{1}$$

so that only a few objects can have small complexity. Conversely, Let μ be a *computable* probability distribution, i.e., such that there is an effective procedure that, given x, computes $\mu(x)$ to any degree of accuracy. Let $K(\mu)$ be the length of the smallest such program. Then,

$$K(x) \leq -\log\mu(x) + K(\mu) + c\ , \tag{2}$$

with c a universal constant. Put $d(x \mid \mu) = -\log\mu(x) - K(x)$. By (1), $t(x \mid \mu) = 2^{d(x \mid \mu)}$ is a payoff function. We now can beat any fraudulent casino. We propose the payoff function $2^{-\log P_n(x) - K(x \mid n)}$. (We use conditional complexity $K(x \mid n)$ because the uniform distribution P_n depends on n.) If every other coin flip comes up heads, then $K(x \mid n) \leq (n/2)+c_0$, and hence we win $2^{t(x)} \geq c_1 2^{n/2}$ from the casino ($c_0, c_1 > 0$), even though the bet does not refer to 'heads'.

The fact that $t(x \mid \mu)$ is a payoff function, implies by Markov's Inequality that for any $k > 0$,

$$\mu\{x : K(x) < -\log\mu(x) - k\} < 2^{-k}. \tag{3}$$

Together, (2) and (3) say that with large probability, the complexity $K(x)$ of a random outcome x is close to its upper bound $-\log\mu(x) - K(\mu)$. If an outcome x violates any 'laws of probability', then the complexity $K(x)$ falls far below the upper bound. Indeed, a proof of some law of probability (like the law of large numbers, the law of iterated logarithm, etc.) always gives rise to some simple computable payoff function $t(x)$ taking large values on the outcomes violating the law. In general, the payoff function $t(x \mid \mu)$ is *maximal* (up to a multiplicative constant) among all payoff functions that are semicomputable (from below). Hence the quantity $d(x \mid \mu)$ constitutes a universal test of randomness - it measures the *deficiency* of randomness in the outcome x with respect to distribution μ, or the extend of justified suspicion against hypothesis μ given the outcome x.

2.9. A Priori Probability

The incomputable 'distribution' $\mathbf{m}(x) = 2^{-K(x)}$ has the remarkable property that the test $d(x \mid \mathbf{m})$ shows all outcomes x random with respect to it. We can interpret (2) and (3) as saying that if the real distribution is μ, then $\mu(x)$ and $\mathbf{m}(x)$ are close to each other with large probability. Therefore, if x comes from some unknown simple distribution μ, then we can use $\mathbf{m}(x)$ as an estimate for $\mu(x)$. Accordingly, Solomonoff has called $\mathbf{m}$ 'a priori probability.' The randomness test $d(x \mid \mu)$ can be interpreted in the framework of hypothesis testing as the likelihood ratio between hypothesis μ and the fixed alternative hypothesis $\mathbf{m}$. In ordinary statistical hypothesis testing, some properties of an unknown distribution μ are taken for granted, and the role of the universal test can probably be reduced to some tests that are used in statistical practice. However, such conditions do not hold in general as is witnessed by prediction of time series in economics, pattern recognition or inductive inference (see below).

Since the a priori probability $\mathbf{m}$ is a good estimate for the actual probability, we can use the conditional a priori probability for prediction - without reference to the unknown distribution μ. For this purpose, we first define a priori probability M for the set of infinite sequences of natural numbers as in [143]. For any finite sequence x, $M(x)$ is the a priori probability that the outcome is some extension of x. Let x, y be finite sequences. Then

$$\frac{M(xy)}{M(x)} \tag{4}$$

is an estimate of the conditional probability that the next terms of the outcome will be given by y provided that the first terms are given by x. It converges to the actual conditional probability $\mu(xy)/\mu(x)$ with μ-probability 1 for any computable distribution μ [123]. Inductive inference formula (4) can be viewed as a mathematical formulation of Occam's razor: predict by the simplest rule fitting the data. The a priori distribution M is incomputable, and the main problem of inductive inference can perhaps be stated as 'finding efficiently computable optimal approximations to M.'

3. Applications of Compressibility

It is not surprising that some strings can be compressed arbitrary far. Easy examples are the decimal expansions for some transcendental numbers like π=3.1415 $\cdots$ and e=2.7 $\cdots$. These strings can be described in $O(1)$ bits, and have therefore constant Kolmogorov complexity. A moment's reflection suggests that the set of computable numbers, i.e., the real numbers computable by Turing machines which start with a blank tape, coincides *precisely* with the set of real numbers of Kolmogorov complexity $O(1)$. This is made precise in the complete version of this paper. Perhaps the most surprising application of the extreme compressibility of some strings is offered by Chaitin's well-known version of Gödel's celebrated incompleteness theorem.

3.1. Chaitin's Version of Gödel's Theorem

Recall K. Gödel's famous incompleteness result that each formal mathematical system which contains Arithmetic is either inconsistent or contains theorems which cannot be proved in the system. In Chaitin's approach [21, 22, 27] we view a theorem - a true statement - together with the description of the formal system, as a description of a proof of that theorem. Just as certain numbers can be really far compressed, like π or 10^{100}, in their descriptions, in a formal mathematical system the ratio between the length of the theorems and the length of their shortest proofs can be enormous. In a sense, Chaitin shows that the worst-case such ratio expressed as a function of the length of the theorem increases faster than any computable function.

More precisely, *although most numbers are random, only finitely many of them can be proved random within a given consistent axiomatic system.* In particular, a system F whose axioms and rules of inference require about k bits to describe cannot be used to prove the randomness of any number much longer than k bits. If the system could prove randomness for a number much longer than k bits, the *first* such proof (first in an unending enumeration of all proofs obtainable by repeated application of axioms and rules of inference) could be used to derive a contradiction: an approximately k-bit program to find and print out the specific random number mentioned in this proof, a number whose smallest program is by assumption considerably larger than k bits. Therefore, even though most strings are random, we will never be able to explicitly exhibit a string of reasonable size which demonstrably possesses this property.

Example. For simplicity of the argument, we use self-delimiting Kolmogorov complexity.
(a) Let theory F be describable in k bits: $K(F) \le k$.
(b) Assume that all true F-expressible statements can be proved in F.
(c) Let $S_c(x)$ be the statement: "x is the lexicographically least binary string of length c with $K(x) \ge c$," expressible in F. Here x is a formal parameter and c an explicit constant, so $K(S_c) \le 2 \log c + O(1)$.

For each c, there exists an x such that $S_c(x) = \mathbf{true}$ is a true statement by a simple counting argument. Moreover, S_c expresses that this x is unique. It is easy to see that combining the descriptions of F, S_c, and this discussion, we obtain a description of this x. Namely, by (b), for each candidate string y of length c, we can decide $S_c(y) = \mathbf{true}$ (holds for $y = x$) or not$(S_c(y)) = \mathbf{true}$ (holds for $y \neq x$), by simple enumeration of all proofs in F. By (a) and (c) therefore $K(x) \leq k + 2\log c + O(1)$ which is a contradiction from some c onward.

As Chaitin expresses it: "... if one has ten pounds of axioms and a twenty-pound theorem, then that theorem cannot be derived from those axioms." Recently, Chaitin has strengthened these results [29].

Example (Levin). Without going into the subtle details, Levin's argument in [70] takes the form that the information $I(\alpha:\beta)$ in a string α about a string β cannot be significantly increased by either algorithmic or probabilistic means. In particular, it leads to the a thesis that "contradicts the assertion of some mathematicians that the truth of any valid proposition can be verified in the course of scientific progress by means of nonformal methods (to do so by formal methods is impossible by Gödel's theorem.)" (For a continuation of this research, see [72].)

3.2. Inductive Inference in Theory Formation

This application stood at the cradle of Kolmogorov complexity proper. It led Solomonoff to formulate the important notion of *a priori* probability, as described in another Section. Solomonoff's proposal [122] is a synthesis of Occam's principle and Turing's theory of effective computability applied to inductive inference. His idea is to view a theory as a compact description of past observations together with predictions of future ones. The problem of theory formation in science is formulated as follows. The investigator observes increasingly larger initial segments of an infinite binary sequence. We can consider the infinite binary sequence as the outcome of an infinite sequence of experiments on some aspect X of Nature. To describe the underlying regularity of this sequence, the investigator tries to formulate a theory that governs X, on the basis of the outcome of past experiments. Candidate theories are identified with computer programs that compute infinite binary sequences starting with the observed initial segment.

To make the discussion precise, such computer programs consist of inputs of a fixed universal Turing machine. The length of the shortest input which yields a particular output string is invariant between a pair of universal machines, up to a constant which depends on these universal machines alone. This observation allows us to attribute a particular Kolmogorov complexity to each individual string without reference to extraneous devices.) To predict future observations, any of the theories is *a priori* equally likely. The investigator needs a criterion to choose among the candidate theories. Applying Occam's razor "entities should not be multiplied beyond necessity," Solomonoff prefers the candidate with the shortest program. See for his computational heuristics [123].

The notion of complexity of equivalent theories as the length of a shortest program that computes a given string emerges forthwith, and also the invariance of this measure under changes of computers that execute them. The metaphor of Natural Law being a compressed description of observations is singularly appealing. Among others, it gives substance to the view that a Natural Law is better if it "explains" more, that is, describes more observations. On the other hand, if the sequence of observations is sufficiently random, then it is subject to no Law but its own description. This metaphorical observation was also made by Chaitin [19].

A related notion has been applied in the recently developed Valiant learning model [13, 129], as in the next Section.

3.3. Learnability in the Valiant Learning Model

This Section is rather closely related to the previous Section. In the previous Section, the (precise) inductive inference is based on infinite input. Obviously if we are to *precisely* infer a nature's law, such an infinite (or exponential) behavior is inherent. However, for the purpose of machine learning, it is sufficient to just *learn* such a law *approximately*: if a human child (or a computer) would recognize, with .99 *probability*, the next apple after seeing three apples, we consider that the concept of apple is learned. In 1983, Valiant [129] introduced such learning model. For simplicity and convenience, we consider the problem learning Boolean formulae of n variables.

According to Valiant, a concept F is a Boolean formula. Those vectors v such that $F(v)=1$ are called positive examples, the rest are negative examples of F. For any F, there are many possible boolean formulae f such that f is consistent with the concept F. Let $|f|$ denote the smallest number of symbols needed to write the representation f. The learning algorithm has available two buttons labeled POS and NEG. If POS (NEG) is pushed, a positive (negative) example is generated according to some fixed but unknown probability distribution D^+ (D^-) according to nature. We assume nothing about the distributions D^+ and D^- except that $\Sigma_{f(v)=1} D^+(v)=1$ and $\Sigma_{f(v)=0} D^-(v)=1$. Let A be a class of concepts. Then A is learnable from examples iff there exists a polynomial p and a (possibly randomized) learning algorithm L such that, for f in A and $\varepsilon>0$, algorithm L halts in $p(n, |f|, 1/\varepsilon)$ time and examples, and outputs a formula $g \in A$ that with probability at least $1-\varepsilon$ has the following properties: $\Sigma_{g(v)=0} D^+(v) < \varepsilon$ and $\Sigma_{g(v)=1} D^-(v) < \varepsilon$.

Many classes of concepts are shown to be learnable in Valiant's sense [13, 47, 56, 81, 110, 129, 130]. (See [55] for a survey.) In [13], again by the Occam's principle, it was shown that given a set of positive and negative data, any consistent concept of size "reasonably" less than the size of data is an "approximately" correct concept. That is, if one can find a shorter representation of data, then one learns. The shorter the conjecture is, the more and better it explains with higher probability. Interestingly, the similar principle of Occam's razor was proposed by Rissanen in 1978 (independently), known as Minimum Description Length Principle [109]. Quinlan and Rivest used this principle to construct an algorithm for constructing decision trees and the result was quite satisfactory compared to existing algorithms [107].

3.4. The Number of Wisdom Ω

This Kabbalistic exercise follows Chaitin [21,22] and Bennett [9]. A real is *normal* if each digit from 0 to 9, and each block of digits of equal length, occurs with equal asymptotic frequency. No rational number is normal to any base, and almost all irrational numbers are normal to every base. But for particular ones, like π and e, it is not known whether they are normal, although statistical evidence suggests they are. In contrast to the randomly appearing sequence of the decimal representation of π, the digit sequence of Champernowne's number 012345678910111213... is very nonrandom yet provably normal [30]. Once we know the law that governs π's sequence, we can make a fortune betting at fair odds on the continuation of a given initial segment, and most gamblers would eventually win against Champernowne's number because they will discover its law.

Almost all real numbers are Kolmogorov random, which implies that no possible betting strategy, betting against fair odds, can win infinite gain. Can we exhibit a specific such number? One can define an uncomputable number $K=0.k(1)k(2)\cdots$, such that $k(i)=1$ if the ith program in a fixed enumeration of programs for some fixed universal machine halts, else $k(i)=0$. By the unsolvability of the halting problem, K is noncomputable. However, by Barzdin's Theorem (before), K is *not* incompressible: each n-length prefix K_n of K can be compressed to a string of length not more than $2\log n$ (since $K(K_n \mid n)\leq\log n$), from some n onwards. It is also easy to see that a gambler can still make infinite profit, by betting only on solvable cases of the halting problem, of which there are infinitely many. Chaitin [23] has found a number that is random in the strong sense needed.

First, fix your favorite universal computer which uses self-delimiting programs of 0's and 1's. Ω equals the probability that a universal computer halts when its program is generated by fair coin tosses. (I.e., the summed *a priori* probabilities of all strings in Solomonoff's sense, cf. above.) Then, Ω is a number between 0 and 1. It is Kolmogorov random, it is noncomputable, and no gambling scheme can make an infinite profit against it. It has the curious property that it encodes the halting problem very compactly. Namely, suppose we want to determine whether a program p halts or not. Let program p have length n. Its probability in terms of coin tosses is 2^{-n}. If we know the first n bits Ω_n of Ω, then $\Omega_n<\Omega\leq\Omega_n+2^{-n}$. However, dovetailing (execute phases 1,2,..., with phase i consists of executing one step of each of the first i programs) the running of all programs sufficiently long, must yield eventually an approximation Ω' of Ω with $\Omega'>\Omega_n$. If p is not among the halted programs which contributed to Ω', then p will never halt, since otherwise its contribution would yield $\Omega>\Omega'+2^{-n}$, which is a contradiction. (I.e., Ω_n is a short program to obtain K_m with $m\approx 2^n$.)

Therefore, knowing the first 10,000 bits of Ω enables us to solve the halting of all programs of less than 10,000 bits. This includes programs looking for counter examples to Fermat's Last Theorem, Riemann's Hypothesis and most other conjectures in mathematics that can be refuted by single finite counter examples. Moreover, for all axiomatic mathematical theories which can be expressed compactly enough to be conceivably interesting to human beings, say in less than 10,000 bits, $\Omega_{10,000}$ can be used to decide for every statement in the theory whether it is true, false or independent. Finally, knowledge of Ω_n suffices to determine whether $K(x)\leq n$ for each finite binary string x. Thus, Ω is truly the number of Wisdom, and ''can be known of, but not known, through human reason'' [9].

3.5. Computable Numbers are Not Random

One can make precise the off-hand claim above that the set of computable numbers coincides with the set of reals which have Kolmogorov complexity $O(1)$. Let $N=\{0,1,\cdots\}$ be the set of natural numbers, let $S=\{\epsilon,0,1,01,10,11,\cdots\}$ be the set of finite binary strings, and let X be the set of infinite binary strings. We denote by $|s|$ the length of a string s, and by s_n the prefix of length n of a string s. (If $x\in X$ then $|x|=\infty$.) An infinite string x is *recursive* iff there is a recursive function $f: N\rightarrow S$ such that $x_n=f(n)$ for all n. It can be shown [24] that x is recursive iff there exists a constant $c>0$ such that for all $n\in N$ we have $K(x_n)\leq K(n)+c$.

4. Application in Mathematics: Weak Prime Number Theorems

Using Kolmogorov complexity, it is easy to derive a weak version of the prime number theorem. An adaptation of the proof Chaitin gives of Euclid's theorem that the number of primes is infinite [26] yields a very simple proof of a weak prime number theorem. Let $\pi(n)$ denote the number of prime numbers less than n. We prove that $\pi(n)$ is $\Omega(\log n\,(\log\log n)^{-1})$. Let n be a random number with $K(n)\geq\log n - O(1)$. Consider a prime factorization

$$n = p_1^{e_1} \cdot p_2^{e_2} \cdots p_m^{e_m},$$

with $p_1, p_2, \cdots$ the sequence of primes in increasing order. With $m=\pi(n)$, we can describe n by the $\pi(n)$ length vector of exponents $(e_1, \ldots, e_{\pi(n)})$. Since $p_i \geq p_1=2$, it holds $e_i\leq\log n$, and, bounding $K(e_i)$ by the length of self-delimiting descriptions of e_i, $K(e_i)\leq\log\log n+2\log\log\log n$, for all $i\leq m$. Therefore, $K(n)\leq\pi(n)(\log\log n + 2\log\log\log n)$. Substituting the lower bound on $K(n)$, we obtain the claimed lower bound on $\pi(n)$ for the special sequence of random n. Observing that π is monotonic increasing and, for each large enough n, the overwhelming majority of strings of length $\log n$ is random, extends the validity of the claimed lower bound to all positive integers n.

Recently, P. Berman [Personal Communication] obtained the stronger result that the number of primes below n is $\Omega(n/(\log n(\log\log n)^2))$, by an elementary Kolmogorov complexity argument. The following proof is based on Berman's unpublished note to us. It is interesting mainly because it shows a relation between primality and prefix codes. (There are other simple elementary methods to obtain weak prime number theorems.) Assume that we have a function $C:N\rightarrow N$ with the following property: for every two integers m,n, $\text{bin}(C(m))$ is not a prefix of $\text{bin}(C(n))$. Consider only C such that $C(n)=o(n^2)$. (E.g., choose $\text{bin}(C(n))$ the self-delimiting description $\text{bin}(|\text{bin}(n)|)\text{bin}(n)$ of n with at most $\log n + 2\log\log n$ bits.)

Lemma (Berman). *For an infinite subsequence of positive integers n, $C(n)=\Omega(p_n)$, where p_n is the nth prime.*

Proof. Assume the contrary, i.e., $C(n)=o(p_n)$. Then, for each $c>0$, there exists a m_1, such that for all $i>m_1$ we have $p_i>c\,C(i)$. By choice of C there also exists a m_2, such that for all $i>m_2$ we have $i^2>c\,C(i)$. Choose m_1,m_2 as large as needed ($m_1 \gg m_2$) to make the following true. For each $n\geq M$, with $M=(m_1!)m_2$, such that either (a) or (b) holds (not necessarily exclusive):
(a) For some $i>m_1$, $p_i \mid n$ (i.e., n is divided by a "large" prime).
(b) For some $i>m_2$, $i^2 \mid n$ (i.e., n is divided by a "large" square).

Fix a Kolmogorov random string x such that (considering $1x$ as the corresponding integer) $1x>M$. We give a short coding for $n=1x$:
(i) If $p_i \mid n$ for some $i>m_1$, the code of n is $0\text{bin}(C(i))\text{bin}(n/p_i)$;
(ii) if $i^2 \mid n$ for some $i>m_2$, the code of n is $1\text{bin}(C(i))\text{bin}(n/i^2)$.

Either way, the length of the coding is at least $\log c$ less than $|x|$. Hence $K(x)\leq|x|-\log c$, contradiction. □

Now we only need an efficient coding for $C(n)$. The self-delimiting Kolmogorov complexity described in a previous Section gives us a coding for $C(n)$ with a loss of $2\log\log n$ additive factor in coding length. Hence $C(n)\leq n\log^2 n$. Therefore, by the Lemma, p_n is $O(n\log^2 n)$. This implies $\pi(n)$ is $\Omega(n/\log^2 n)$. For C an order 2 description, $|\text{bin}(C(n))| \leq \log n + \log\log n + 2\log\log\log n$, we similarly obtain p_n is $O(n\log n(\log\log n)^2)$, and $\pi(n)$ is $\Omega(n/(\log n(\log\log n)^2))$.

Can we strengthen Berman's result in a simple way? Clearly, it can be readily extended to $\pi(n)$ is $\Omega(n/(\log n\log\log n(\log\log\log n)^2))$, for infinitely many n, using order 3 self-delimiting encodings, and so on.

5. Application of Incompressibility: Proving Lower Bounds

It was observed in [103], that static, descriptional (program size) complexity of a *single* random string can be used to obtain lower bounds on dynamic, computational (running time) complexity. The power of the static, descriptional Kolmogorov complexity in the dynamic, computational lower bound proofs rests on one single idea: There are incompressible (or Kolmogorov random) strings. In a traditional lower bound proof by counting, it usually involves *all* inputs (or all strings of certain length) and one shows that the lower bound has to hold for *some* of these ("typical") inputs. However since a particular "typical" input is *hard* to construct, the proof has to involve all the inputs. Now we understand that a "typical input" can be constructed via a Kolmogorov random string. However, as we have shown in relation with Gödel's Theorem, we will never be able to put our hands on one of those strings or inputs and claim that it is random or "typical". No wonder the old counting arguments had to involve all inputs, it was because a particular typical input cannot be *proved* to be "typical" or random. In a Kolmogorov complexity proof, we choose a random string that *exists.* That it cannot be exhibited is no problem, since we only need existence. As a routine, the way one proves a lower bound by Kolmogorov complexity is as follows: Fix a Kolmogorov random string which we know exists, even though we do not have the concrete string in hand. Prove the lower bound with respect to this particular *fixed* string: show that if the lower bound does not hold, then this string can be compressed. Because we are dealing with only one fixed string, the lower bound proof usually becomes quite easy and natural.

In the next sub-section, we give three examples to illustrate the basic methodology. In the following subsections, we survey the lower bound results obtained using Kolmogorov complexity of the past 10 years (1979-1988). Many of these results resolve old or new, some of them well-known, open questions; Some of these results greatly simplify and improve the existing proofs.

5.1. Three Examples of Proving Lower Bounds

In this section, we illustrate how Kolmogorov complexity is used to prove lower bounds by three concrete examples.

Example 1 (One Tape Turing Machines).

Consider a most basic Turing machine model with only one tape, with a 2-way read/write head, which serves as both input and work tape. The input is initially put on the first n cells of the only tape. We refer a reader who is not familiar with Turing machines to [50] for a detailed definition. The following theorem was first proved by Hennie and a proof by counting, for comparison, can be found in [50], page 318. [101] presented the following elegant proof. Historically, this was the first lower bound obtained by Kolmogorov-complexity.

Theorem. *It requires $\Omega(n^2)$ steps for the above single tape TM to recognize $L=\{ww^R : w \in \{0,1\}^*\}$ (the palindromes).*

Proof [101]. Assume on the contrary, M accepts L in $o(n^2)$ time. Fix a Kolmogorov random string w of length n for a large enough n. Consider the computation of M on ww^R. A *crossing sequence* associated with a tape square consists of the sequence of (state, time) pairs for which the tape head crosses the intersquare boundary between this square and its left neighbor. If *c.s.* is a crossing sequence, then $|c.s.|$ denotes the length of its description. Divide the tape segment containing ww^R into three equal length segments. If each crossing sequence associated with a square in the middle segment is longer than $\frac{n}{10|M|}$ then M spent $\Omega(n^2)$ time on this input. Otherwise there is a crossing sequence of length less than $\frac{n}{10|M|}$. Assume that this occurs at c_0. Now this crossing sequence requires at most $n/10$ bits to encode. W.l.o.g. assume c_0 is left of the middle. Using this crossing sequence, we reconstruct w as follows: For every string x of length n, put x on the input tape and start to simulate M. Each time when the head reaches c_0 from the left, we take the next element in the crossing sequence to skip the computation of M when the head is on the right of c_0 and resume the simulation starting from the time when the head moves back to the left of (or on) c_0 again. If the simulation ends consistently, *i.e.* every time the head moves to c_0 the current status of M is consistent with that specified in the crossing sequence, then $w=x$. Since if $w \neq x$ then M accepts a wrong input xw^R. However this implies

$$K(w) < |c.s.| + O(\log n) < n,$$

contradicting to $K(w) \geq n$. □

Example 2 (Parallel Addition).

Consider the following widely used and most general parallel computation model, priority PRAM. A priority PRAM consists of processors $P(i)$ $i=1,2,\cdots,n^{O(1)}$, and an infinite number of shared memory cells $C(i)$, $i=1,2,\cdots$. Each step of the computation consists of three parallel phases as follows. Each processor: (1) reads from a shared memory cell, (2) performs a computation, and (3) may attempt writing into some shared memory cell. At each step each processor is in some *state*. The actions and the next state of each processor at each step depend on the current state and the value read. In case of *write conflicts*, the processor with the minimum index succeeds in writing.

Theorem. *Adding n integers, each of polynomial number of bits, requires $\Omega(\log n)$ parallel steps on a priority PRAM.*

Remark. A weaker version than the one above was first proved in [48] using a Ramsey theorem, and in [54, 99]. In these references one needs to assume that the integers have arbitrarily many bits, or exponentially many bits. A more precise version of the above theorem was proved in [77]. Beame [8] obtained a different proof, independently.

Proof [77]. Suppose that a priority PRAM M with $n^{O(1)}$ processors adds n integers in $o(\log n)$ parallel steps for infinitely many n's. The programs (maybe infinite) of M can be encoded into an oracle A. The oracle, when queried about (i,l), returns the initial section of length l of the program for $P(i)$. Fix a string $X \in \{0,1\}^{n^3}$ such that $K^A(X) \geq |X|$. Divide X equally into n parts $x_1, x_2, \cdots, x_n$. Then consider the (*fixed*) computation of M on input $(x_1, \cdots, x_n)$. We inductively define (with respect to X) a processor to be *alive* at step t in this computation if

(1) it writes the output; or

(2) it succeeds in writing something at some step $t' \geq t$ which is read at some step $t'' \geq t'$ by a processor who is alive at step t''.

An input is *useful* if it is read at some step t by a processor alive at step t. By simple induction on the step number we have: for a T step computation, the number of useful inputs and the number of processors ever alive are both $O(2^T)$.

It is not difficult to see that, given all the useful inputs and the set $ALIVE = \{(P(i), t_i) : P(i)$ was alive until step $t_i > 0\}$, we can simulate M to uniquely reconstruct the output $\sum_{i=1}^{n} x_i$. Since $T = o(\log n)$, we know $2^T = o(n)$. Hence there is an input x_{i_0} which is not useful. We need $O(2^T \log P) = o(n \log n)$ bits to represent $ALIVE$. To represent $\{x_i : i \neq i_0\}$ we need $n^3 - n^2 + \log n$ bits, where $\log n$ bits are needed to indicate the index i_0 of the missing input. The total number bits needed in the simulation is less than

$$J = n^3 - n^2 + O(n \log n) + O(\log n) < n^3.$$

But from these J bits we can find $\sum_{i=1}^{n} x_i$ by simulating M using the oracle A, and then reconstruct x_{i_0} from $\sum_{i=1}^{n} x_i$ and $\{x_i : i \neq i_0\}$. But then

$$K^A(X) \leq J < n^3.$$

This contradicts the randomness of X. □

Example 3 (Parallel Computing of a Minimum Index).

Consider a different PRAM model with n processors $P(1), \cdots, P(n)$ and with only one memory cell, $C(1)$. The write-conflicts are resolved by requiring that if several processors attempt to write into $C(1)$ at the same time, then they must all write the same thing. Each processor possesses one input bit. The problem is to find the smallest index i such that $P(i)$ has input bit 1. This is a basic problem for understanding the differences between several commonly used PRAM models. An $\Omega(\log n)$ lower bound was proved before by [37] by a fixing bit method. What we are interested in here is a Kolmogorov complexity proof.

Suppose M is a parallel machine of the above type and M solves our problem in $d = o(\log n)$ steps. Choose a K^A-random string X of $\log n$ bits with respect to the oracle A which contains all the programs of

$P(1), P(2), \cdots, P(n)$. Consider the following input to M: $I=0^{X-1}1^{n-X+1}$. We will construct a d bit vector v while simulating M on I as follows: At the ith step of the simulation, $v(i)=1$ if a processor with input 1 succeeds in writing, $v(i)=0$ if a processor with input 0 succeeds in writing, and $v(i)=2$ if no processor writes at step i. Now we use only A, v to reconstruct X. Assume that the contents of $C(1)$ at steps $1, \cdots, i-1$ are already known (to each processor) by induction. Simulate M as follows: at step i, if $v(i)=0$ [$v(i)=1$] then for $j=1,2,\cdots$ simulate $P(j)$ [$P(n-j+1)$], on input zero [one], i steps to see if it writes at the ith step, and stop once we find first j such that $P(j)$ [$P(n-j+1)$] writes. The value written will be the contents of $C(1)$ at step i because whenever several processors write at the same time they write the same thing. We are also guaranteed with the correct value by the construction of vector v. If $v(i)=2$, then at step i nobody wrote, so skip this step in the simulation and the contents of $C(1)$ at step i is unchanged. Hence for $i=d$ we must have X written in $C(1)$ which implies $K^A(X)=|v|=o(\log n)$, which is a contradiction.

Remark. Note that a lower bound obtained by Kolmogorov complexity usually implies that the lower bound holds for "almost all strings". This is the case for all three examples. In this sense the lower bounds obtained by Kolmogorov complexity are usually stronger than those obtained by its counting counterpart, since it usually also implies directly the lower bounds for nondeterministic or probabilistic versions of the considered machine. We will discuss this later.

5.2. Lower Bounds: More tapes versus fewer tapes

Although Barzdin [7] and Paul [101] are the pioneers of using Kolmogorov complexity to prove lower bounds, the most influential paper is probably the one by Paul, Seiferas and Simon [103], which was presented at the 1980 STOC. This was partly because [101] was not widely circulated and, apparently, the paper by Barzdin [7] did not even reach this community. The major goal of [103] was "to promote the approach" of applying Kolmogorov complexity to obtain lower bounds. In [103], apart from other results, the authors with the aid of Kolmogorov complexity, remarkably simplified the proof of a well-known theorem of Aanderaa [1]: *real-time* simulation of k tapes by $k-1$ tapes is impossible for deterministic Turing machines.

In this model the Turing machine has k (work) tapes, apart from a separate input tape and (possibly) a separate output tape. This makes the machine for each $k \geq 1$ far more powerful than the model of Example 1, where the single tape is both input tape and work tape. E.g., a 1-(work)tape Turing machine can recognize the palindromes of Example 1 in real-time $T(n)=n$ in contrast with $T(n)=\Omega(n^2)$ required in Example 1.

At the 1982 Paul [104], using Kolmogorov complexity, extended the results in [103] to: on-line simulation of real-time $k+1$-tape Turing machines by k-tape Turing machines requires $\Omega(n(\log n)^{1/(k+1)})$ time.

To simulate k tapes with 1 tape, the known (and trivial) upper bound on the simulation time was $O(n^2)$. Paul's lower bound decreased the gap with this upper bound only slightly. But in later developments w.r.t. this problem Kolmogorov complexity has been very successful. The second author, not using Kolmogorov complexity, reported in [134] a $\Omega(n^{1.5})$ lower bound on the time to simulate a single pushdown store on-line by one *oblivious* tape unit. However, using Kolmogorov complexity the technique worked also without the oblivious restriction, and yielded in quick succession [136, 137], and the optimal results cited hereafter. Around 1983/1984, independently and in chronological order*, Wolfgang Maass at UC Berkeley, the first author at Cornell and the second author at CWI Amsterdam, obtained a square lower bound on the time to simulate two tapes by one tape (deterministically), and thereby closed the gap between 1 tape versus k (w.l.o.g. 2) tapes. These lower bounds, and the following ones, were proven with the simulator an *off-line* machine with *one-way* input. All three relied on Kolmogorov complexity, and actually proved more in various ways*. Thus, Maass also obtained a nearly optimal result for nondeterministic simulation: [86] exhibits a language that can be accepted by two deterministic one-head tape units in real-time, but for which a one-head tape unit requires $\Omega(n^2)$ time in the deterministic case, and $\Omega(n^2/(\log n)^2 \log\log n)$ time in the nondeterministic case. This lower bound was later improved by [79] to $\Omega(n^2/\log n \log\log n)$ time using Maass' language, and by Galil, Kannan, and Szemeredi [40] to $\Omega(n^2/\log^{(k)} n)$ (for any k, with $\log^{(k)}$ is the k-fold iterated logarithm) by an ingenious construction of a language whose computation graph does not have small separators. This almost closed the gap in the nondeterministic case. In their final combined paper, Li and Vitányi [79] presented the following lower bounds, all by Kolmogorov complexity. To simulate 2 pushdown stores, or only 1 queue, by 1 deterministic tape requires $\Omega(n^2)$ time. Both bounds are tight. (Note that the 2 pushdown store result implies the 2 tape result. However, the 1

* *Historical note.* A claim for an $\Omega(n^{2-\epsilon})$ lower bound for simulation of two tapes by both one deterministic tape and one nondeterministic tape was first circulated by W. Maass in August 1983, but did not reach Li and Vitányi. Maass submitted his extended abstract containing this result to STOC by November 1983, and this did not reach the others either. The final STOC paper of May 1984 (submitted February 1984) contained the optimal $\Omega(n^2)$ lower bound for the deterministic simulation of two tapes by one tape. In M. Li: 'On 1 tape versus 2 stacks,' Tech. Rept. TR-84-591, Dept. Comp. Sci., Cornell University, January 1984, the $\Omega(n^2)$ lower bound was obtained for the simulation of two pushdown stores by one deterministic tape. In: P.M.B. Vitányi, 'One queue or two pushdown stores take square time on a one-head tape unit,' Tech. Rept. CS-R8406, Centre for Mathematics and Computer Science, Amsterdam, March 1984, the $\Omega(n^2)$ lower bound was obtained for the simulation of two pushdown stores (or the simulation of *one* queue) by one deterministic tape. Maass's and Li's result were for off-line computation with one-way input, while Vitányi's result was for on-line computation.

queue result is incomparable with either of them.) Further, 1-tape nondeterministic simulation of two pushdown stores requires $\Omega(n^{1.5}/\sqrt{\log n})$ time. This is almost tight because of [75]. Finally, 1-tape nondeterministic simulation of one queue requires $\Omega(n^{4/3}/\log^{2/3} n)$ time. The corresponding upper bound of the last two simulations are $O(n^{1.5}\sqrt{\log n})$ in [75]. In a successor paper, together with Luc Longpré, we have extended the above work with a comprehensive study stressing queues in comparison to stacks and tapes [78]. There it was shown that a queue and a tape are not comparable, i.e. neither can simulate the other in linear time. Namely, simulating 1 pushdown store (and hence 1 tape) by 1 queue requires $\Omega(n^{4/3}/\log n)$, in both the deterministic and nondeterministic cases. Simulation of 1 queue by 1 tape was resolved above, and simulation of 1 queue by 1 pushdown store is trivially impossible. Nondeterministic simulation of 2 queues (or 2 tapes) by 1 queue requires $\Omega(n^2/(\log^2 n \log\log n))$ time, and deterministic simulation of 2 queues (or 2 tapes) by 1 queue requires quadratic time. All these results would be formidable without Kolmogorov complexity.

A next step is to attack the similar problem with a *2-way input* tape. Maass and Schnitger [87] proved that when the input tape is 2-way, 2 work tapes are better than 1 for *computing a function* (in contrast to recognizing a language). The model is a Turing machine with no output tape; the function value is written on the work tape(s) when the machine halts. It is interesting to note that they considered a matrix transposition problem, as considered in Paul's original paper. Apparently, in order to transpose a matrix, a lot of information needs to be shifted around which is hard for a single tape. [87] showed that transposing a matrix (with element size $O(\log n)$) requires $\Omega(n^{3/2}(\log n)^{-1/2})$ time on a 1-tape off-line Turing machine with an extra 2-way read-only input tape. The first version of this paper (single authored by Maass) does not actually depend on Kolmogorov complexity, but has a cumbersome proof. The final Kolmogorov complexity proof was much easier and clearer. (This lower bound is also optimal [87].) This gives the desired separation of two tape versus one, because, with two work tapes, one can sort in $O(n \log n)$ time and hence do matrix transposition in $O(n \log n)$ time. Recently, Maass, Schnitger, and Szemeredi [88] in 1987 finally resolved the question of whether 2 tapes are better than 1 with 2-way input tape, for *language recognition*, with an ingenious proof. The separation language they used is again related to matrix transposition except that the matrices are Boolean and sparse (only $\log^{-2} n$ portion of nonzeros): $\{A{**}B : A=B^t$ and $a_{ij} \neq 0$ only when $i,j = 0 \bmod (\log m)$ where m is the size of matrices$\}$. The proof techniques used combinatorial arguments rather than Kolmogorov complexity. There is still a wide open gap between the $\Omega(n \log n)$ lower bound of [88] and the $O(n^2)$ upper bound. In [87] it was observed that if the Turing machine has a 1-way output tape on which the transposed matrix can be written, transposition of Boolean matrices takes only $O(n^{5/4})$. Namely, with only one work tape and no output tape, once some bits have been written they can be moved later only by time wasting sweeps of the work tape head. In contrast, with an output tape, as long as the output data are computed in the correct order they can be output and don't have to be moved again. Using Kolmogorov complexity, in [36] Dietzfelbinger shows that transposition of Boolean matrices by Turing machines with 2-way input tape, 1 work tape, and a 1-way output tape requires $\Omega(n^{5/4})$ time, thus matching the upper bound for matrix transposition.

5.3. Lower Bounds: More heads versus fewer heads

Again applying Kolmogorov complexity, Paul [102] showed that 2-dimensional 2 tape (with one head on each tape) Turing machines cannot on-line simulate 2-dimensional Turing machines with 2 heads on 1 tape in real time. He was not able to resolve this problem for 1-dimensional tapes, and, despite quite some effort, the following problem is open and believed to be difficult: Are two (1-dimensional) tapes, each with one head, better than 2 heads on one (1-dimensional) tape? The following result, proved using Kolmogorov complexity, is intended to be helpful in separating these classes. A Turing machine with two 1-head storage tapes cannot simulate a queue in both real time and with at least one storage head always within $o(n)$ squares from the start square [135]. (Thus, most prefixes of the stored string need to be shifted all the time, while storing larger and larger strings in the simulator, because the simulator must always be ready to reproduce the stored string in real-time. It would seem that this costs too much time, but this has not been proved yet.) To eventually exploit this observation to obtain the desired separation, J. Seiferas [118] proved the following 'equal information distribution' property. For no c (no matter how large) is there a function $f(n)=o(n)$, such that every sufficiently long string x has a description y with the properties: $|y|=c|x|$, and if x' is a prefix of x and y' is any subword of y with $|y'|=c|x'|$ then $K(x'|y')<f(K(x))$.

Multihead finite automata and pushdown automata were studied in parallel with the field of computational complexity in the years of 1960's and 1970's. One of the major problems on the interface of the theory of automata and complexity is to determine whether additional computational resources (heads, stacks, tapes, etc.) increase the computational power of the investigated machine. In the case of multihead machines it is natural to ask whether $k+1$ heads are better than k. A k-head finite (pushdown) automaton is just like a finite (pushdown) automaton except having k 1-way heads on the input tape. Two rather basic questions were left open from the automata and formal language theory of 1960's:

(1) Rosenberg Conjecture (1965): $(k+1)$-head finite automata are better than k-head finite automata [112, 113].

(2) Harrison-Ibarra Conjecture (1968): $(k+1)$-head pushdown automata are better than k-head pushdown automata. Or, there are languages accepted by $(k+1)$-DPDA but not k-PDA [44].

In 1965, Rosenberg [113] claimed a solution to problem (1), but Floyd [38] pointed out that Rosenberg's informal proof was incomplete. In 1971 Sudborough [124, 125], and later Ibarra and Kim [53] obtained a partial solution to problem (1) for the case of 2 heads versus 3 heads, with difficult proofs. In the 1976 Yao and Rivest [142] finally presented a full solution to problem (1). A different proof was also obtained by Nelson [96]. Recently it was noted by several people, including Joel Seiferas and the authors, that the Yao-Rivest proof can be done very naturally and easily by Kolmogorov complexity: Let

$$L_b = \{w_1\# \cdots \#w_b \$ w_b\# \cdots \#w_1 : w_i \in \{0,1\}^* \}.$$

as defined by Rosenberg and Yao-Rivest. Let $b = \left[\frac{k}{2} \right] + 1$. So L_b can be accepted by a $(k+1)$-DFA. Assume that a k-FA M also accepts L_b. Let W be a long enough Kolmogorov random string and W be equally partitioned into $w_1 w_2 \cdots w_b$. We say that the 2 w_i's in L_b are matched if there a time such that 2 heads of M are in the 2 w_i's concurrently. Hence there is an i such that w_i is not matched. Then apparently, this w_i can be generated from $W - w_i$ and the positions of heads and states for M when a head comes in/out w_i, $K(w_i \mid W - w_i) = O(k \log n) < |w_i|/2$, contradiction.

The HI-conjecture, however, was open until the time of Applied Kolmogorov complexity. Several authors tried to generalize the Yao-Rivest method [94, 95] or the Ibarra-Kim method [31] to the k-PDA case, but only partial results were obtained. For the complete Odyssey of these efforts see the survey in [32]. With the help of Kolmogorov complexity, [32] presented a complete and transparent solution to the Harrison-Ibarra conjecture for the general case. The proof was constructive, natural, and quite simple compared to the partial solutions. The basic idea, ignoring the technical details, was generalized from the above proof we gave for the Rosenberg conjecture.

A related problem of whether a k-DFA can do string-matching was raised by Galil and Seiferas [39] They proved that a 6-head 2-way DFA can do string-matching, *i.e.*, accept $L = \{x\#y : x \text{ is a substring of } y\}$. In 1982, when the first author and Yaacov Yesha, then at Cornell, tried to solve the problem, we achieved a difficult and tediously long proof (many pages), by counting, that 2-DFA cannot do string matching. Later J. Seiferas suggested the use of Kolmogorov complexity, which shortened the proof to less than a page [76]! By similar methods a proof that 3-DFA cannot do string matching was also obtained [74].

5.4. Lower Bounds: Parallel computation and Branching-programs

In Examples 2 and 3 we saw that the remarkable concept of Kolmogorov complexity does not only apply to lower bounds in restricted Turing machines, it also applies to lower bounds in other general models, like parallel computing models and branching-programs.

Fast addition or multiplication of n numbers in parallel is obviously important. In 1985 Meyer auf der Heide and Wigderson [48] proved, using Ramsey theorems, that on priority PRAM, the most powerful parallel computing model, ADDITION (and MULTIPLICATION) requires $\Omega(\log n)$ parallel steps. Independently, a similar lower bound on addition was obtained by Israeli and Moran [54] and Parberry [99]. All these lower bounds depend on inputs from infinite (or exponentially large) domains. However, in practice, we are often interested in small inputs. For example, addition of n numbers of $n^{1/\log\log n}$ bits each can be done in $O(\log n / \log\log n)$ time with $n^{O(1)}$ processors which is less than the $\Omega(\log n)$ lower bound of [48]. In 1986 we [77] applied Kolmogorov-complexity to obtain parallel lower bounds (and tradeoffs) for a large class of functions with arguments in small domains (including Addition, Multiplication ...) on priority PRAM. As a corollary, for example, we show that for numbers of polynomial size, it takes $\Omega(\log n)$ parallel steps for addition. This improved the results of [48, 54, 99]. Furthermore the proof is really natural and intuitive, rather than the complicated counting as before. Independently, Paul Beame at the same meeting also obtained similar results, but using a different partition method. A proof of the above result was given in Example 2.

As another example, we prove a depth 2 unbounded fan-in circuit requires $\Omega(2^n)$ gates from {AND,OR,NOT} to compute the parity function: Assume the contrary. Let C be a binary encoding of integer n and such a circuit with $o(2^n)$ gates. W.l.g., let the first level of C be AND gates and the second level be an OR gate. Consider an $x = x_1 \cdots x_n$ such that $K(x \mid C) \ge |x| = n$ and $PARITY(x) = 1$. Now, any AND gate of fan-in at least n must be 0 since otherwise we can specify x by the index of that gate which is $o(\log_2 2^n)$. Therefore, since $PARITY(x) = 1$ some AND gate, G, of fan-in less_than n must be 1. Then G includes neither x_i nor $\bar{x}_i$ for some i. Hence changing only the value of x_i in x does not change the output (value 1) of G and C, contradiction. (Note, more careful calculation on the constants can result in a more precise bound.)

Sorting is one of the most studied problems in computer science, due to its great practical importance. (As we have seen it was also studied by Paul in [101].) In 1979 Borodin, Fischer, Kirkpatrick, Lynch, and Tompa proved a time-space trade-off for comparison based sorting algorithms [16]. This was improved and generalized to a very wide class of sequential sorting algorithms by Borodin and Cook [17] defined as 'branching programs.' The proof involved difficult counting. In [108] Reisch and Schnitger used Kolmogorov complexity, in one of their three applications, to greatly simplify the well-known $\Omega(n^2/\log n)$ bound of Borodin and Cook [17] for the time-space trade-off in sorting with branching-programs. They also improved the lower bound in [17] to $\Omega(n^2 \log\log n / \log n)$.

5.5. Lower Bounds: Very Large Scale Integration

It should not be surprising that Kolmogorov complexity can be applied to VLSI lower bounds. Many VLSI lower bounds were based on the crossing sequence type arguments similar to that of Turing machines [80]. This sort of arguments can be readily converted to much more natural and easier Kolmogorov complexity arguments like the one used in Example 1.

We use the model of Lipton and Sedgewick [80], which is a generalization of Thompson's Model [126]. All lower bounds proved here also apply to the Thompson model. Roughly speaking there are three main components in the model: (a) The (n-input, 1 output) Boolean function $f(x_1,x_2, \cdots ,x_n)$ which is to be computed; (b) A synchronous circuit C, that computes f, which contains *and, or, not* gates of arbitrary fan-in and fan-out and with n fixed input gates (i.e., what is called where-oblivious) that are not necessarily on the boundary of the layout of C (the time an input arrives may depend on the data value); (c) And a VLSI (for convenience: rectangle) layout V that realizes C, where wires are of unit width and processors occupy unit squares. A central problem facing the VLSI designers is to find C that computes a given f in time T and a VLSI layout of C with area A, minimizing say AT^2 as introduced by Thompson [126] and later generalized by [80].

This method used to prove $AT^2=\Omega(n^2)$ lower bounds for many problems was roughly as follows: Draw a line to divide the layout into two parts, with about half inputs on each part. Suppose the line cuts through ω wires, then $A>\Omega(\omega^2)$. Further, since for each time unit only one bit of information can flow through a wire, $T>I/\omega$ where I is the *amount* of information that has to be passed between the 2 parts. Then for each specific problem one only needs to show that $I=\Omega(n)$ for any division. Lipton and Sedgewick defined *crossing sequence* to be, roughly, the sequence of T tuples $(v_1, \cdots ,v_\omega)$ where the ith tuple contains the values appeared at the cut of width ω at step i.

Now it is trivial to apply our Kolmogorov complexity to simplify the proofs of *all* VLSI lower bounds obtained this way. Instead of complicated and non-intuitive counting arguments which involves *all* inputs, we now demonstrate how easy one can use one single Kolmogorov random string instead. The lower bounds before the work of [80] were for n-input and n-output functions, the Kolmogorov complexity can be even more trivially applied there. We only look at the harder n-input 1-output problems stated in [80]. A sample question in [80]:

Example (Pattern Matching). *Given a binary text string of* $(1-\alpha)n$ *bits and a pattern of* αn *bits, with* $\alpha<1$, *determine if the pattern occurs in the text.*

Proof Sketch. Let C implement pattern matching with layout V. Consider any cut of V of width ω which divides inputs into 2 halves. Now it is trivial that $I=\Omega(n)$ since for a properly arranged Kolmogorov random text and pattern this much information must pass the cut. This finishes the proof of $AT^2 \geq \Omega(n^2)$. □

All other problems, Selection/Equality testing, DCFL, Factor Verification, listed in [80] can all be done similarly, even under the nondeterministic, or randomized circuits as defined in [80].

Some general considerations on VLSI lower bounds using Kolmogorov complexity were given by R. Cuykendall [34]. L.A. Levin and Y.Y. Itkis have informed us about their work in progress on the VLSI computation model under different information transmission assumptions, using Kolmogorov complexity [71]. In their model, if the speed of information transmission is superlinear, namely $\max(K(d)-\log f(d))<\infty$ for $f(d)$ the time for a signal to traverse a wire of length d, then a chip can be simulated by a chip in which all long wires have been deleted (which results in a considerable savings in required area). Note that $f(d)=\Omega(d \log^2 d)$ suffices, but not $f(d)=O(d)$.

5.6. Lower Bounds: Randomized Algorithms

We have seen that Kolmogorov complexity can be naturally applied to nondeterministic Turing machines. It is almost certain that it is useful for analyzing randomized algorithms. Indeed this is the case. In their paper about three applications of Kolmogorov complexity [108] Reisch and Schnitger analyzed, using Kolmogorov complexity, the probabilistic routing algorithm in n-dimensional cubes of Valiant and Brebner [128].

In 1983 Paturi and Simon generalized the deterministic lower bounds previously proved by [1, 102-104] etc., to probabilistic machines. This is based on the following elegant idea (based on a note of, and discussions with, R. Paturi): As we mentioned before, all the Kolmogorov complexity proofs depend on only a fixed Kolmogorov random string α. If the lower bound fails, then this incompressible string can be compressed, hence a contradiction. [100] proved a version of the Symmetry of Information Lemma we stated in a previous section. They show that for a sequence of random coin tossing, the probability that this sequence of random coin tossing bits, β, contains much information about α is vanishingly small. Observe that if α is Kolmogorov random relative to the coin tossing sequence β, then the old deterministic argument would just fall through with β as an extra useless input (or oracle as in example 1). And note that many such α exists. Hence, (ignoring technical details) using this idea and careful construction of the input for the probabilistic simulator, it was shown that, on the average, the probabilistic simulator would not give any advantage in reducing the computation time.

Remark. Similar ideas were expressed earlier by Levin who called the general principle involved "Law of Information Conservation" [70]. See for later developments also [72].

5.7. Lower Bounds: Formal Language Theory

The classic introduction to formal language theory is [50]. An important part of formal language theory is deriving a hierarchy of language families. The main division is the Chomsky hierarchy, with regular languages, context-free languages, context-sensitive languages and recursively enumerable languages. The common way to prove that certain languages are not regular [not context-free] is by using "pumping" lemma's, i.e., the *uvw*-lemma [*uvwxy*-lemma]. However, these lemma's are complicated to state and cumbersome to prove or use. In contrast, below we show how to replace such arguments by simple, intuitive and yet rigorous, Kolmogorov complexity arguments. We present some unpublished material from our paper in preparation [73]. W.l.o.g., languages are infinite sets of strings over a finite alphabet.

Regular languages coincide with the languages accepted by finite automata (FA). Another way of stating this is by the Myhill-Nerode Theorem: each regular language over alphabet V consists of the union of some equivalence classes of a right-invariant equivalence relation on V^* $(=\bigcup_{i\geq 0} V^i)$ of finite index. Let us give an example of how to use Kolmogorov complexity to prove non-regularity. We prove that $\{0^k 1^k : k \geq 1\}$ is not regular. To the contrary, suppose it is regular. Fix k with $K(k) \geq \log k$, with k large enough to derive the contradiction below. The state q of the accepting FA after processing 0^k is, up to a constant, a description of k. Namely, by running the FA, starting from state q, on a string consisting of 1's, it reaches its first accepting state precisely after k 1's. Hence, there is a constant c, depending only on FA, such that $\log k < c$, which is a contradiction. We generalize this observation, actually a Kolmogorov-complexity interpretation of the Myhill-Nerode Theorem, as follows. (In lexicographic order short strings precede long strings.)

Lemma (KC-Regularity). *Let L be regular. Then for some constant c depending only on L and for each string x, if xy is the n th string in the lexicographical order in* $\{xy : xy \in L\}$ *then* $K(y) \leq K(n) + c$.

Proof. Let L be a regular language. A string y such that $xy \in L$, for some x and n as in the Lemma, can be described by

(a) This discussion, and a description of the FA that accepts L,

(b) The state of the FA after processing x, and the number n. □

The KC-regularity lemma can be applied whenever the pumping lemma can be applied. It turns out that the converse of our Lemma also holds. Therefore, the above Lemma also applies to situations when the normal pumping lemma(s) do(es) not apply. Further it is easier and more intuitive than pumping lemmas. For example:

Example. We prove that $\{1^p : p \text{ is prime}\}$ is not regular. Consider the string xy consisting of p 1's, with p is the $(k+1)$th prime. Set in the lemma x equal to $1^{p'}$ with p' the kth prime, so $y = 1^{p-p'}$, and $n=1$. It follows that $K(p-p') = O(1)$. Since the differences between the consecutive primes rise unbounded (by the prime number theorem, or as easy consequence of Euclid's proof below), this implies that there is an unbounded number of integers of Kolmogorov complexity $O(1)$. Since there are only $O(1)$ descriptions of length $O(1)$, we have a contradiction. (A simple way to argue that $p-p'$ rises unbounded is as follows. Let P be the product of the first j primes. Clearly, no $P+i$, $1 \leq i \leq j$, is prime.)

Example [Exercise 3.1(h^*) **in [50]].** Prove that $L = \{xx^R w : x, w \in \{0,1\}^*\}$ is not regular. Fix x such that $K(x) \geq |x|$. Consider prefix $(01)^{3\log|x|} x$. The first string with this prefix in L is $(01)^{3\log|x|} x x^R (10)^{3\log|x|} 0$. By KC-regularity lemma, $K(x^R (10)^{3\log|x|} 0) \leq K(1) + c$, contradiction.

Example [Exercise 3.6* **in [50]].** Prove that $L = \{0^i 1^j : GCD(i,j) = 1\}$ is not regular. Obviously L is regular iff $L' = \{0^i 1^j : GCD(i,j) \neq 1\}$ is regular. Fix a prime p such that $K(p) \geq \log p - \log\log p$ (by density of primes). Consider prefix 0^p. By the KC-regularity lemma, $K(1^p) \leq K(2) + c$, a contradiction.

5.8. Lower Bounds: Unproved ones

This section summarizes the open questions we consider to be important and may be solvable by Kolmogorov complexity.

(1) Can k-DFA do string-matching [39]?

(2) Are 2 heads on one (1-dimensional) tape better than two (1-dimensional) tapes each with one head?

(3) Prove tight, or $\Omega(n^{1+\epsilon})$, lower bound for simulating two tapes by one for off-line Turing machines with an extra 2-way input tape.

6. Resource-Bounded Kolmogorov Complexity and Its Applications

Here we treat several notions of resource-bounded Kolmogorov complexity, with applications ranging from the P=NP question to factoring integers and cryptography. Several authors suggested early on the possibility of restricting the power of the device used to compress strings. Says Kolmogorov [62]

> ''The concept discussed ... does not allow for the ''difficulty'' of preparing a program p for passing from an object x to an object y. ... [some] object permitting a very simple program, i.e., with very small complexity $K(x)$ can be restored by short programs only as the result of computations of a thoroughly unreal nature. ... [this concerns] the relationship between the necessary complexity of a program and its permissible difficulty t. The complexity $K(x)$ that was obtained [before] is, in this case, the minimum of $K^t(x)$ on the removal of the constraints on t.''

The earliest use of resource-bounded Kolmogorov complexity we know of is Barzdin's 1968 result

[7] cited earlier. Time-limited Kolmogorov complexity was applied by Levin [68] in relation with his independent work on NP-completeness, and further studied in [72]. Adleman investigated such notions [2], in relation to factoring large numbers. Resource-bounded Kolmogorov complexity was investigated by Daley [35], Ko [59] and Huynh [52] who prove several results of recursion-theoretic flavor. Sipser [120] used time-limited Kolmogorov complexity to show that the class BPP (problems which can be solved in polynomial time with high probability) is contained in the polynomial time hierarchy: $BPP \leq \Sigma_4 \cap \Pi_4$. (Gács improved this to $BPP \leq \Sigma_2 \cap \Pi_2$.) We treat the approaches of Adleman, Bennett and Hartmanis in more detail below. Let us note here that there is some relation between the approaches to resource-bounded Kolmogorov complexity by Adleman [2], Levin [72], and Bennett [10].

6.1. Potential

In an elegant paper [2], Adleman formulates the notion of *potential* as the amount of time that needs to be pumped in a number by the computation that finds it. Namely, while constructing a large composite number from two primes we spend only a small amount of time. However, to find the primes back may be difficult and take lots of time. Is there a notion of storing potential in numbers with the result that high-potential primes have relatively low-potential products? Such products would be hard to factor, because all methods must take the time to pump the potential back. Defining the appropriate notion, Adleman shows that if factoring is not in P then this is the reason why. Formally,

For all integer $k \geq 0$, for all $x \in \{0,1\}^*$ (for all $y \in \{0,1\}^*$), x *is* k-potent (with respect to y) iff there is a program p of size $\leq k\,|\mathrm{bin}(|x|)|$, which with blanks ($y$) as input halts with output x in less than or equal to $|x|^k$ steps. (Recall that $|x|$ is the length of x and $\mathrm{bin}(n)$ is the usual binary representation, without nonsignificant zeros, of a positive integer n.)

Example. For almost all $n \in N$, 1^n is 2-potent. Namely, $|1^n| = n$ and $|\mathrm{bin}(n)| \sim \log n$. Then it is not difficult to see that, for each large enough n, there is a program p, $|p| < 2\log n$, that computes 1^n in less than n^2 steps.

Example. For all k, for almost all incompressible x, x is not k-potent. This follows straightaway from the definitions.

Example. Let u be incompressible. If $v = u + 1^{666}$, where "+" denotes "exclusive or", then v is incompressible, but also v is 1-potent with respect to u.

Lemma (Adleman). *For $x \in \{0,1\}^*$ and $k \in N$, define $y = y(1)y(2) \cdots y(2^n)$ such that, for $1 \leq i \leq 2^n$, $y(i)=1$ if $\mathrm{bin}(i)$ is k-potent with respect to x and $y(i)=0$ otherwise. Let x' be the self-delimiting version of x. Then, for all k, the function $f_k(x' 1^n) = y$ is computable in polynomial time.*

Proof. There are at most $2 \cdot 2^{k|\mathrm{bin}(n)|} \sim 2n^k$ programs of length $\leq k\,|\mathrm{bin}(n)|$. By simulating all such programs (one after the other) on input x for at most n^k steps the result is obtained. □

We informally state two results proved by Adleman.

Theorem (Adleman). *Factoring is difficult iff multiplication infinitely often takes highly potent numbers and produces relatively low potent numbers.*

Theorem (Adleman). *With respect to the $P=NP$ question: $SAT \in NP - P$ iff for all k there exist infinitely many $\phi \in SAT$ such that [for all T, if truth assignment T satisfies ϕ then T is not k-potent w.r.t. ϕ]*

6.2. Logical Depth

C. Bennett has formulated an intriguing notion of logical depth [10, 11]. Kolmogorov complexity helps to define individual information and individual randomness. It can also help to define a notion of ''individual computational complexity'' of a finite object. Some objects are the result of long development (=computation) and are extremely unlikely to arise by any probabilistic algorithm in a small number of steps. Logical depth is the necessary number of steps in the deductive or causal path connecting an object with its plausible origin. Concretely, the time required by a universal computer to compute an object from its maximally compressed description. Formally, (in P. Gács' reformulation, using the Solomonoff-Levin approach to *a priori* probability):

$$depth_\varepsilon(x) = \min\{t : \mathrm{Prob}_t(x)/\mathrm{Prob}_\infty(x) \geq \varepsilon\}$$

Thus, the depth of a string x is at least t with confidence $1-\varepsilon$ if the conditional probability that x arises in t steps *provided it arises at all* is less than ε. (One can also formulate logical depth in terms of shortest programs and running times [10], or Example below.) According to Bennett, quoted in [25]: ''A structure is deep, if it is superficially random but subtly redundant, in other words, if almost all its algorithmic probability is contributed by slow-running programs. ... A priori the most probable explanation of 'organized information' such as the sequence of bases in a naturally occurring DNA molecule is that it is the product of an extremely long biological process.''

Example (Bennett). Bennett's original definition: Fix, as usual, an optimal universal machine U. A string $x \in \{0,1\}^*$ is logical (d,b)-deep, or ''d-deep at confidence level 2^{-b}'', if every program to compute x in time $\leq d$ is compressible by at least b bits.

The notion is intended to formalize the idea of a string for which the null hypothesis that it originated by an effective process of fewer than d steps, is as implausible as tossing b consecutive heads. Depth should be stable, i.e., no trivial computation should be able to transform a shallow object into a deep one.

Theorem (Bennett). Deep strings cannot quickly be computed from shallow ones. More precisely, There is a polynomial $p(t)$ and a constant c, both depending on U, such that, if x is a program to compute y in time t, and if x is less than (d,b)-deep, then y is less than $(d+p(t), b+c)$-deep.

Example (Bennett). Similarly, depth is reasonably machine independent. If U,U' are two optimal universal machines, then there exists a polynomial $p(t)$ and a constant c, both depending on U,U', such that $(p(d), b+c)$-depth on either machine is a sufficient condition for (d,b)-depth on the other.

Example (Bennett). The distinction between depth and information: consider the numbers K and Ω (see Section on Ω). K and Ω encode the same information, viz. solution to the halting problem. But K is deep and Ω shallow. Because Ω encodes the halting problem with maximal density (the first 2^n bits of K can be computed from the first $n+O(\log n)$ bits of Ω) it is recursively indistinguishable from random noise and practically useless: the time required to compute an initial segment of K from an initial segment of Ω increases faster than any computable function. Namely, Barzdin' [7] showed that the initial segments of K are compressible to the logarithm of their length if unlimited time is allowed for decoding, but can only be compressed by a constant factor if any recursive bound is imposed on the decoding time.

6.3. Generalized Kolmogorov Complexity

Below we partly follow [45]. Assume that we have fixed a universal Turing machine U with an input tape, work tapes and an output tape. A string x is computed from a string z (z is a *description* of x) means that U started with z on its input tape halts with x on its output tape.

Remark. In order to be accurate in the reformulations of notions in the Examples below, we shall assume w.l.o.g. that the set of programs for which U halts is an effective prefix code: no such program is the prefix of any other such program. I.e., we use self-delimiting descriptions as described in a previous section.

In the following we distinguish the main parameters we have been able to think of: compression factor, time, space, and whether the computation is inflating or deflating. A string x has *resource bounded* Kolmogorov complexity ${}_U^{UP}(K,T,S)$ if x can be computed from a string z, $|z| \le K \le |x|$, in $\le T$ steps by U using $\le S$ space on its work tape. A string x of length n is in complexity class $K_U^{UP}[k(n),t(n),s(n)]$ if $K \le k(n)$, $T \le t(n)$ and $S \le s(n)$. Thus, we consider a computation that *inflates* z to x. A string x has *resource bounded* Kolmogorov complexity ${}_U^{DOWN}(K,T,S)$ if some description z of x can be computed from x, $|z| \le K \le |x|$, in $\le T$ steps by U using $\le S$ space on its work tape. Here we consider a computation that *deflates* x to z. A string x of length n is in complexity class $K_U^{DOWN}[k(n),t(n),s(n)]$ if $K \le k(n)$, $T \le t(n)$ and $S \le s(n)$. Clearly,

$$K_U^{DOWN}[k(n),\infty,\infty] = K_U^{UP}[k(n),\infty,\infty] = K[k(n)],$$

$k(n)$ fixed up to a constant, with $K[k(n)]$ is (with some abuse of notation) the class of binary strings x such that $K(x) \ge k(n)$. (Here we denote by K the self-delimiting Kolmogorov complexity).

It is follows immediately by the Hennie-Stearns simulation of many work tapes by two work tapes, that there is a U with two work tapes such that, for any multitape universal Turing machine V, there is a constant c such that

$$K_V^{UP}[k(n),t(n),s(n)] \subseteq$$
$$K_U^{UP}[k(n)+c, c \cdot t(n) \log t(n)+c, c\, s(n)+c]$$

Thus, henceforth we drop the subscripts because the results we derive are invariant up to such small perturbations. It is not difficult to prove, however, that larger perturbations of the parameters separate classes. For instance,

$$K^{UP}[\log n,\infty,n^2] \subset K^{UP}[\log n,\infty,n^2 \log n]$$
$$K^{UP}[\log n,\infty,n^2] \subset K^{UP}[2\log n,\infty,n^2]$$

The obvious relation between inflation and deflation is:

$$K^{UP}[k(n),t(n),\infty] \subseteq K^{DOWN}[k(n),t(n)\,2^{k(n)},\infty],$$
$$K^{DOWN}[k(n),t(n),\infty] \subseteq K^{UP}[k(n),t(n)\,2^n,\infty],$$

(there are at most $2^{k(n)}$ [2^n] possibilities to try).

In his Ph.D. thesis [83], Longpré analysed the structure of the different generalized Kolmogorov complexity sets, with different time and space bounds (the UP version). Longpré builds the resource hierarchies for Kolmogorov complexity in the spirit of classical time and space complexity hierarchies. He related further structural properties to classical complexity. He also extended Martin-Löf's results to generalized Kolmogorov complexity: the space bounded Kolmogorov complexity random strings pass all statistical tests which use less space than the space bound. Finally, he shows how to use Kolmogorov randomness to build a pseudo-random number generator that passes Yao's test [141].

Example (Potency). Adleman's potency [2], can now be reformulated as: $x \in \{0,1\}^*$, $|x|=n$, is k-potent if $x \in K^{UP}[k \log n, n^k, \infty]$.

Example (Logical Depth). Bennett's logical (b,d)-depth [10], can be weakly characterized by: If $x \in \{0,1\}^*$ is logical (d,b)-deep then $x \in K^{UP}[K(x)+b, d, \infty]$. It is not known whether the converse implication holds.

Example (Time of Computation). Related to the notions potential and logical depth is Levin's concept of *time of computation complexity Kt* [72]. In this framework we formulate it by: $x \in \{0,1\}^*$, has Kt-complexity $Kt(x)=m$ if $x \in K^{UP}[m-\log t, t, \infty]$, m minimal.

Example (Hartmanis). The sparse set

$$SAT \cap K^{UP}[\log n, n^2, \infty]$$

is a Cook-complete set for all other sparse sets in NP.

In [45] these and similar results are derived for PSPACE and sets of other densities. It is also used to give new interpretations to oracle constructions, and to simplify previous oracle constructions. This leads to conditions in terms of Kolmogorov complexity under which there exist NP complete sets that are not polynomial-time isomorphic, as formulated in [12]. In [46] a characterization of the P=NP question is given in terms of time-bounded Kolmogorov complexity and relativization. Earlier, Adleman with [2] established a connection, namely, NP $\neq$ P exactly when NP machines can ''manufacture'' randomness. Following this approach, Hemachandra [49] obtains unrelativized connections in the spirit of [46].

Example (Hartmanis). Hartmanis noticed the following interesting fact: A polynomial machine cannot compute from simple input complicated strings and hence cannot ask complicated questions to an oracle A. Using this idea, he constructed several very elegant oracles. As an example, we construct the Baker-Gill-Solovay oracle A such that $P^A \neq NP^A$: By diagonalization, choose $C \subseteq \{1^{2^n} : n \geq 1\}$ and $C \in DTIME[n^{\log n}]-P$. For every n such that $1^{2^n} \in C$ put the first string of length 2^n from

$$K^{UP}[\log n, n^{\log n}, \infty] - K^{UP}[\log n, n^{\log\log n}, \infty]$$

in A. Clearly, $C \in NP^A$. But C cannot be in P^A since in polynomial time, a P^A-machine cannot ask any question about any string in A. Hartmanis also constructed two others including a random sparse oracle A such that $NP^A \neq P^A$ with probability 1.

6.4. Generalized Kolmogorov Complexity Applied to Structural Proofs

Generalized Kolmogorov complexity turns out to be an elegant tool for studying the structure in complexity classes. The first such applications are probably due to Hartmanis, as we discussed in previous section. Other recent work in this area includes [4, 6, 105]. In this section we try to present some highlights of the continuing research in this direction. We will present several excellent constructions, and describe some constructions in detail.

Example (An Exponentially Low Set Not In P). a set A is exponentially low if $E^A = E$, where $E = DTIME[2^{cn}]$. Book, Orponen, Russo, and Watanabe [15] constructed an exponentially low set A which is not in P. We give this elegant contruction in detail. Let $K = K^{UP}[n/2, 2^{3n}, \infty]$. Let $A = \{x : x$ is the lexicographically least element of $\overline{K}$ of length $2^{2^{\cdot^{\cdot^{2}}}}$ (stack of m 2's), for some $m > 0\}$. Obviously $A \in E$. Further A is not in P since otherwise we let $A = L(M)$ and for $|x| \gg |M|$ and $x \in A$ we would have that $x \in K$, a contradiction. We also need to show that $E^A = E$. To simulate a computation of E_A by an E machine: If a query to A is of correct length (stack of 2's) and shorter than cn for a small constant c. Then just do exhaustive search to decide. Otherwise, the answer is "no" since (1) a string of wrong length (no stack of 2's) is not in A and (2) a string of length greater than cn is not even in $\overline{K}$. (2) is true since the query string can be calculated from the input of length n and the exponentially shorter previous queries, which can be encoded in, say, $cn/4$ bits assuming c chosen properly, therefore the query string is in K.

In [139], Watanabe used time-space bounded Kolmogorov complexity to construct a more sophisticated set D which is polynomial Turing complete for E but not complete for E under polynomial truth-table reduction. Allender and Watanabe [5] used Kolmogorov complexity to characterized the class of sets which are polynomial many-one equivalent to tally sets, in order to study the question of whether $E_m^P(Tally) = E_{btt}^P(Tally)$ is true, where $E_{=}^P(Tally) = \{L :$ for some tally set T, $L \equiv^P T\}$. In [51, 52], Huynh started a series of studies on the concept of resource-bounded Kolmogorov complexity of languages. He defined that the (time-/space-bounded) Kolmogorov complexity of a language to be the (time-/space-bounded) Kolmogorov complexity of $Seq(L^{<n}) = C_L(w_1)C_L(w_2) \cdots C_L(w_{2^n-1})$, where w_i is the lexicographically the ith word and $C_L(w_i) = 1$ iff $w_i \in L$. In particular, he shows that there is a language $L \in DTIME(2^{2^{o(n)}})$ (any hard set for this class) such that the 2^{poly}-time-bounded Kolmogorov complexity of L is exponential almost everywhere. *I.e.*, the sequence $Seq(L^{<n})$ cannot be compressed to a subexponentially short string within 2^{poly} time for all but finitely many n's. Similar results were also obtained for space-bounded classes. He used these results to classify exponential-size circuits.

6.5. A Kolmogorov Random Reduction

The original ideas of this Section belong to U. Vazirani and V. Vazirani [131]. We re-formulate their results in more natural and simpler terms of Kolmogorov complexity.

In 1979 Adleman and Manders defined a probabilistic reduction, called UR-reduction, and showed several number-theoretic problems to be hard for NP under UR-reductions but not known to be NP-hard. In [131] the notion is refined as follows:

> A is PR-reducible to B, denoted as $A \leq_{PR} B$, iff there is a probabilistic polynomial time TM T and $\delta > 0$ such that (1) $x \in A$ implies $T(x) \in B$, and (2) x not in A implies Prob($T(x)$ not in B)$\geq \delta$. A problem is *PR−complete* if every NP problem can be PR-reduced to it.

Vazirani and Vazirani obtained the first non number-theoretic PR-complete problem, which is still not known to be NP-complete up to today: ENCODING BY TM:

INSTANCE:Two strings $x, y \in \{0,1,2,\alpha,\beta\}^*$, integer k.

QUESTION:Is there a TM M with k or fewer states that on input x generates y in $|y|$ steps. (M has one read-write tape initially containing x and a write-only tape to write y. M must write one symbol of y each step, *i.e.* real-time.)

PR−completeness Proof. We reduce ENCODING BY FST to our problem, where the former is NP-complete and is defined as:

INSTANCE:Two strings $x, y \in \{0,1,2\}^*$, $|x| = |y|$, and integer k.

QUESTION:Is there a finite state transducer M with k

or less states that outputs y on input x. (Each step, M must read a symbol and output a symbol.)

Reduction: any instance (x,y,k) of ENCODING BY FST is transformed to (xr,yr,k) for ENCODING BY TM, where $K(r \mid x,y) \geq |r| - c_\delta$, and $r \in \{\alpha,\beta\}^*$. For a given δ, Prob(generate such an r)$\geq \delta$. Clearly if there is a FST F of at most k state that outputs y on input x, then we can construct a TM with outputs yr on input xr by simply adding two new transitions from each state back to itself on α,β and output what it reads. If there is not such FST, then the k state TM must reverse its read head on prefix x when producing y. Hence it produces r without seeing r (notice the real-time requirement). Hence $K(r \mid x,y) = O(1)$, a contradiction.

7. Conclusion

The opinion has sometimes been voiced that Kolmogorov complexity has only very abstract use. We are convinced that Kolmogorov complexity is immensely useful in a plethora of applications ranging from very theoretic to quite practical. We believe that we have given conclusive evidence for that conviction by this collection of applications.

In our view the covered material represents only the onset of a potentially enormous number of applications of Kolmogorov complexity in mathematics and the sciences. By the examples we have discussed, readers may get the feel how to use this general purpose tool in their own applications, thus turning the next two decades into the golden spring of Kolmogorov complexity.

Acknowledgements

We are grateful to Greg Chaitin, Peter Gács, Leonid Levin and Ray Solomonoff for taking lots of time to tell us about the early history of our subject, and introducing us to many exciting applications. Additional comments were provided by Donald Loveland and Albert Meyer. Juris Hartmanis and Joel Seiferas introduced us in various ways to Kolmogorov complexity. P. Berman, L. Levin and Y. Itkis, and R. Paturi kindly supplied to us (and gave us permission to use) their unpublished material about the prime number theorem, chips without long wires, and lower bounds for probabilistic machines, respectively. O. Watanabe and L. Longpré supplied to us interesting material in connection with the structure of complexity classes. Discussions with Peter van Emde Boas, Jan Heering, Evangelos Kranakis, Ker-I Ko, Michiel van Lambalgen, Lambert Meertens, and Umesh Vazirani are gratefully acknowledged. Th. Tsantilas and J. Rothstein supplied many useful references. We dedicate this work to A.N. Kolmogorov, who unexpectedly died while this paper was being written.

References

1. Aanderaa, S.O., "On k-tape versus (k-1)-tape real-time computation," pp. 75-96 in *Complexity of Computation*, ed. R.M. Karp, American Math. Society, Providence, R.I. (1974).
2. Adleman, L., "Time, space, and randomness," Report MIT/LCS/79/TM-131, Massachusetts Institute of Technology, Laboratory for Computer Science (March 1979).
3. Aleksandrov, P.S., "A few words on A.N. Kolmogorov," *Russian Math. Surveys* **38**, pp. 5-7 (1983).
4. Allender, E., "Some consequences of the existence of pseudorandom generators," Proc. 19th ACM Symposium on Theory of Computing (1987).
5. Allender, E. and O. Watanabe, "Kolmogorov complexity and degrees of tally sets," *Proceedings 3rd Conference on Structure in Complexity Theory* (1988).
6. Balcazar, J. and R. Book, "On generalized Kolmogorov complexity," *Proc. 1st Structure in Complexity Theory Conf., Springer LNCS* **223**, pp. 334-340 (1986).
7. Barzdin', Y.M., "Complexity of programs to determine whether natural numbers not greater than n belong to a recursively enumerable set," *Soviet Math. Dokl.* **9**, pp. 1251-1254 (1968).
8. Beame, P., "Limits on the power of concurrent-write parallel machines," 18th ACM Symp. on Theory of Computing (1986).
9. Bennett, C.H., "On random and hard to describe numbers," Mathematics Tech. Rept. RC 7483 (#32272), IBM Watson Research Center, Yorktown Heights (May 1979).
10. Bennett, C.H., "On the logical "depth" of sequences and the reducibilities to random sequences," Tech. Rept., IBM Watson Research Center, Yorktown Heights (February 1981).
11. Bennett, C.H., "Dissipation, information, computational complexity and the definition of organization," pp. 297-313 in *Emerging syntheses in Science, Proceedings of the Founding Workshops of the Santa Fe Institute*, ed. D. Pines (1985).
12. Berman, L. and J. Hartmanis, "On isomorphisms and density of NP and other complete sets," *SIAM J. on Computing* **6**, pp. 305-327 (1977).
13. Blumer, A., A. Ehrenfeucht, D. Haussler, and M. Warmuth, "Classifying Learnable Geometric Concepts With the Vapnik-Chervonenkis Dimension," Proceedings 18th ACM Symp. on Theory of Computing (1986).
14. Bogolyubov, N.N., B.V. Gnedneko, and S.L. Sobolev, "Andrei Nikolaevich Kolmogorov (on his eightieth birthday)," *Russian Math. Surveys* **38**, pp. 9-27 (1983).
15. Book, R., P. Orponen, D. Russo, and O. Watanabe, "On exponential lowness," *SIAM J. Computing*, (To appear).
16. Borodin, A., M.J. Fischer, D.G. Kirkpatrick, N.A. Lynch, and M. Tompa, "A time-space tradeoff for sorting and related non-oblivious computations," 20th IEEE Symposium on Foundations of Computer Science (1979).
17. Borodin, A. and S. Cook, "A time-space tradeoff for sorting on a general sequential model of computation," 12th ACM Symp. on Theory of Computing (1980).
18. Carnap, R., *Logical Foundations of Probability*, University of Chicago Press, Chicago, Ill. (1950).
19. Chaitin, G.J., "On the length of programs for computing finite binary sequences," *J. Assoc. Comp. Mach.* **13**, pp. 547-569 (1966).
20. Chaitin, G.J., "On the length of programs for computing finite binary sequences: statistical considerations," *J. Assoc. Comp. Mach.* **16**, pp. 145-159 (1969).
21. Chaitin, G.J., "Information-theoretic limitations of formal systems," *J. Assoc. Comp. Mach.* **21**, pp. 403-424 (1974).
22. Chaitin, G.J., "Randomness and mathematical proof," *Scientific American* **232**, pp. 47-52 (May 1975).
23. Chaitin, G.J., "A theory of program size formally identical to information theory," *J. Assoc. Comp. Mach.* **22**, pp. 329-340 (1975).

24. Chaitin, G.J., ''Information-theoretic characterizations of recursive infinite strings,'' *Theor. Comput. Sci.* **2**, pp. 45-48 (1976).

25. Chaitin, G.J., ''Algorithmic Information Theory,'' *IBM J. Res. Dev.* **21**, pp. 350-359 (1977).

26. Chaitin, G.J., ''Toward a mathematical definition of ''life'','' pp. 477-498 in *The Maximal Entropy Formalism*, ed. M. Tribus, MIT Press, Cambridge, Mass. (1979).

27. Chaitin, G.J., ''Godel's theorem and information,'' *International Journal of Theoretical Physics* **22**, pp. 941-954 (1982). (Reprinted in T. Tymoczko, New Directions in the Philosophy of Mathematics, Birkhauser, Boston, 1986.)

28. Chaitin, G.J., *Information, Randomness and Incompleteness - Papers on Algorithmic Information Theory,* World Scientific, Singapore (1987).

29. Chaitin, G.J., *Algorithmic Information Theory,* Cambridge University Press (1987).

30. Champernowne, D.G., ''The construction of decimals normal in the scale of ten,'' *J. London Math. Soc.* **8**, pp. 254-260 (1933).

31. Chrobak, M., ''Hierarchies of one-way multihead automata languages,'' *12th ICALP, Springer LNCS* **194**, pp. 101-110, (Also in Theoretical Computer Science 48 (1986) pp.153-181) (1985).

32. Chrobak, M. and M. Li, ''k+1 heads are better than k for PDAs,'' pp. 361-367 in *27th IEEE FOCS* (1986).

33. Church, A., ''On the concept of a random sequence,'' *Bull. Amer. Math. Soc.* **46**, pp. 130-135 (1940).

34. Cuykendall, R.R., ''Kolmogorov information and VLSI lower bounds,'' Ph.D. Thesis, University of California, Los Angeles (Dec. 1984).

35. Daley, R.P., ''On the inference of optimal descriptions,'' *Theoretical Computer Science* **4**, pp. 301-309 (1977).

36. Dietzfelbinger, M., ''Lower bounds on computation time for various models in computational complexity theory,'' Ph.D. Thesis, University of Illinois at Chicago (1987).

37. Fich, F., P. Ragde, and A. Wigderson, ''Relations between concurrent-write models of parallel computation,'' Proc. 3rd ACM Symp. on Principles of Distributed computing (1984).

38. Floyd, R., ''Review 14,'' *Comput. Rev.* **9**, p. 280 (1968).

39. Galil, Z. and J. Seiferas, ''Time-space optimal matching,'' Proc. 13th ACM Symposium on Theory of Computing (1981).

40. Galil, Z., R. Kannan, and E. Szemeredi, ''On nontrivial separators for k-page graphs and simulations by nondeterministic one-tape Turing machines,'' pp. 39-49 in *Proc. 18th ACM STOC* (1986).

41. Gnedneko, B.V., ''Andrei Nikolaevich Kolmogorov (on the occasion of his seventieth birthday),'' *Russian Math. Surveys* **28**, pp. 5-16 (1973).

42. Gács, P., ''On the symmetry of algorithmic information,'' *Soviet Math. Dokl.* **15**, p. 1477, (Correction, *Ibid.*, 15(1974)) (1974).

43. Gács, P., ''Randomness and probability - complexity of description,'' pp. 551-555 in *Encyclopedia of Statistical Sciences*, ed. Kotz-Johnson, John Wiley & Sons (1986).

44. Harrison, M.A. and O.H. Ibarra, ''Multi-head and multi-tape pushdown automata,'' *Information and Control* **13**, pp. 433-470 (1968).

45. Hartmanis, J., ''Generalized kolmogorov complexity and the structure of feasible computations,'' pp. 439-445 in *Proc. 24th IEEE FOCS* (1983).

46. Hartmanis, J. and L. Hemachandra, ''On sparse oracles separating feasible complexity classes,'' *Proc. 3rd Symposium on Theoretical Aspects of Computer Science, Springer LNCS* **210**, pp. 321-333 (1986).

47. Haussler, D., N. Littlestone, and M Warmuth, ''Expected mistake bounds for on-line learning algorithms,'' Manuscript (1988).

48. Heide, F. Meyer auf der and A. Wigderson, ''The complexity of parallel sorting,'' 17th ACM Symp. on Theory of computing (1985).

49. Hemachandra, L., ''Can P and NP manufacture randomness?,'' Tech. Rept. TR86-795, Dept. Computer Science, Cornell University (December 1986).

50. Hopcroft, J.E. and J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation,* Addison-Wesley (1979).

51. Huynh, D.T., ''Non-uniform complexity and the randomness of certain complete languages,'' Techn. Rept. TR 85-34, Iowa State University (December, 1985.).

52. Huynh, D.T., ''Resource-bounded Kolmogorov complexity of hard languages,'' Proc. 1st Structure in Complexity Theory Conf., Springer LNCS (1986).

53. Ibarra, O.H. and C.E. Kim, ''On 3-head versus 2 head finite automata,'' *Acta Infomatica* **4**, pp. 193-200 (1975).

54. Israeli, A. and S. Moran, *Private communication.*

55. Kearns, M., M. Li, L. Pitt, and L. Valiant, ''Recent results on Boolean concept learning,'' in *Machine Learning*, ed. T. Mitchell. (To appear)

56. Kearns, M., M. Li, L. Pitt, and L. Valiant, ''On the learnability of Boolean formulae,'' Proc. 19th ACM Symp. on Theory of Computing (1987).

57. Knuth, D., *Semi-Numerical Algorithms,* Addison-Wesley (1981). (Second edition)

58. Knuth, D.E., ''Big Omicron and big Omega and big Theta,'' *SIGACT News* 8(2), pp. 18-24 (1976).

59. Ko, Ker-I, *Resource-bounded program-size complexity and pseudorandom sequences,* Department of computer science, University of Houston (1983).

60. Kolmogorov, A.N., *Grundbegriffe der Wahrscheinlichkeitsrechnung,* Berlin (1933). (2nd Russian Edition (1974), ''Osnovnye Poniatija Teorii Verojatnostej,'' Nauka, Moscow)

61. Kolmogorov, A.N., ''On tables of random numbers,'' *Sankhya, The Indian Journal of Statistics, Series A* **25**, pp. 369-376 (1963).

62. Kolmogorov, A.N., ''Three approaches to the quantitative definition of information,'' *Problems in Information Transmission* **1**(1), pp. 1-7 (1965).

63. Kolmogorov, A.N., ''Logical basis for information theory and probability theory,'' *IEEE Trans. on Information Theory* **IT-14.5**, pp. 662-664 (1968).

64. Kolmogorov, A.N., ''Combinatorial foundations of information theory and the calculus of probabilities,'' *Russian Mathematical Surveys* **38**, pp. 29-40 (1983).

65. Lambalgen, M. van, ''Von Mises' definition of random sequences reconsidered,'' *Journal of Symbolic Logic* **52**, pp. 725-755 (1987).

66. Lambalgen, M. van, ''Random Sequences,'' Ph.D. Thesis, Universiteit van Amsterdam, Amsterdam (1987).

67. Laplace, P.S., *A Philosophical Essay on Probabilities,* Dover, New York (1819). (Date is of original publication.)

68. Levin, L.A., "Universal search problems," *Problems in Information Transmission* **9**, pp. 265-266 (1973).

69. Levin, L.A., "On the notion of a random sequence," *Soviet Math. Dokl.* **14**, pp. 1413-1416 (1973).

70. Levin, L.A., "Laws of information conservation (non-growth) and aspects of the foundation of probability theory," *Problems in Information Transmission* **10**, pp. 206-210 (1974).

71. Levin, L.A., "Do chips need wires?," Manuscript/NSF proposal MCS-8304498, Computer Science Department, Boston University (1984(?)).

72. Levin, L.A., "Randomness conservation inequalities; information and independence in mathematical theories," *Information and Control* **61**, pp. 15-37 (1984).

73. Li, M. and P.M.B. Vitányi, "Formal language theory by Kolmogorov complexity," In preparation.

74. Li, M., "Lower Bounds in Computational Complexity," Ph.D. Thesis, Report TR-85-663, Computer Science Department, Cornell University (March 1985).

75. Li, M., "Simulating two pushdowns by one tape in O(n**1.5 (log n)**0.5) time," 26th Annual IEEE Symposium on the Foundations of Computer Science (1985).

76. Li, M. and Y. Yesha, "String-matching cannot be done by 2-head 1-way deterministic finite automata," *Information Processing Letters* **22**, pp. 231-235 (1986).

77. Li, M. and Y. Yesha, "New lower bounds for parallel computation," 18th ACM Symposium on Theory of Computing (1986).

78. Li, M., L. Longpré, and P.M.B. Vitányi, "On the power of the queue," pp. 219-233. in *Structure in Complexity Theory, Lecture Notes in Computer Science* (1986).

79. Li, M. and P.M.B. Vitányi, "Tape versus queue and stacks: The lower bounds," *Information and Computation* **77** (1988).

80. Lipton, R. and R. Sedgewick, "Lower bounds for VLSI," 13th ACM Symp. on Theory of computing (1981).

81. Littlestone, N., "Learning quickly when irrelevant attributes abound: A new linear threshold algorithm," Proc. 28th IEEE Symposium on Foundations of Computer Science (1987).

82. Littlewood, J.E., "The dilemma of probability theory," pp. 71-73 in *Littlewood's Miscellany*, ed. B. Bollobás, Cambridge University Press (1986). (Revised edition)

83. Longpré, L., "Resource bounded Kolmogorov complexity, a link between computational complexity and information theory," Ph.D. Thesis, Techn. Rept. TR-86-776, Computer Science Department, Cornell University (1986).

84. Loveland, D.W., "On minimal-program complexity measures," Proceedings ACM Symposium on Theory of Computing (1969).

85. Loveland, D.W., "A variant of the Kolmogorov concept of complexity," *Information and Control* **15**, pp. 510-526 (1969).

86. Maass, W., "Combinatorial lower bound arguments for deterministic and nondeterministic Turing machines," *Trans. Amer. Math. Soc.* **292**, pp. 675-693 (1985).

87. Maass, W. and G. Schnitger, "An optimal lower bound for Turing machines with one work tape and a two-way input tape," *Structure in Complexity Theory, Lecture Notes in Computer Science* **223**, pp. 249-264 (1986).

88. Maass, W., G. Schnitger, and E. Szemeredi, "Two tapes are better than one for off-line Turing machines," Proc. 19th ACM Symposium on Theory of Computing (1987).

89. Martin-Lof, P., "The definition of random sequences," *Information and Control* **9**, pp. 602-619 (1966).

90. Martin-Lof, P., "Complexity oscilations in infinite binary sequences," *Zeitschrift f:ur Wahrscheinlichkeitstheory und Verwandte Gebiete* **19**, pp. 225-230 (1971).

91. Metropolis, N.C., G. Reitweiser, and J. von Neumann, "Statistical treatment of values of the first 2,000 decimal digits of *e* and π calculated on the ENIAC," in *John von Neumann, Collected Works, Vol. V.*, ed. A.H. Traub, MacMillan, New York (1963).

92. Minsky, M.L., "Problems of Formulation for Artificial Intelligence," pp. 43 in *Mathermatical Problems in the Biological Sciences, Proceedings of Symposia in Applied Mathematics XIV*, ed. R.E. Bellman, American Mathematical Society, Providence, R.I. (1962).

93. Mises, R. von, *Probability, Statistics and Truth*, MacMillan, New York (1939). (Reprint: Dover, 1981)

94. Miyano, S., "A hierarchy theorem for multihead stack-counter automata," *Acta Informatica* **17**, pp. 63-67 (1982).

95. Miyano, S., "Remarks on multihead pushdown automata and multihead stack automata," *Journal of Computer and System Sciences* **27**, pp. 116-124 (1983).

96. Nelson, C.G., "One-way automata on bounded languages," TR14-76, Harvard University (July 1976).

97. Neumann, J. von, "Various techniques used in connection with random digits," in *John von Neumann, Collected Works, Vol. V.*, ed. A.H. Traub, MacMillan, New York (1963).

98. Obituary,, "Mr. Andrei Kolmogorov - Giant of mathematics," *Times* (October 26, 1987).

99. Parberry, I., "A complexity theory of parallel computation," Ph.D. Thesis, Warwick University (1984).

100. Paturi, R. and J. Simon, "Lower bounds on the time of probabilistic on-line simulations," pp. 343 in *Proc. 24th IEEE FOCS* (1983).

101. Paul, W., "Kolmogorov's complexity and lower bounds," Proc. 2nd International Conference on Fundamentals of Computation Theory (September 1979).

102. Paul, W., "On heads versus tapes," *Theoretical Computer Science* **28**, pp. 1-12 (1984).

103. Paul, W.J., J.I. Seiferas, and J. Simon, "An information theoretic approach to time bounds for on-line computation," *J. Computer and System Sciences* **23**, pp. 108-126 (1981).

104. Paul, W.J., "On-line simulation of k+1 tapes by k tapes requires nonlinear time," *Information and Control*, pp. 1-8 (1982).

105. Peterson, G., "Succinct representations, random strings and complexity classes," Proc. 21st IEEE Symposium on Foundations of Computer Science (1980).

106. Popper, K.R., *The Logic of Scientific Discovery*, University of Toronto Press, Toronto (1959).

107. Quinlan, J. and R. Rivest, "Inferring decision trees using the minimum description length principle," Draft, MIT Lab for Computer Science, Cambridge, Mass. (1987).

108. Reisch, S. and G. Schnitger, "Three applications of Kolmogorov-complexity," pp. 45-52. in *Proc. 23rd IEEE FOCS* (1982).

109. Rissanen, J., "Modeling by the shortest data description," *Automatica* **14**, pp. 465-471 (1978).

110. Rivest, R., "Learning Decision-Lists," Unpublished manuscript, MIT Lab. for Computer Science

(December, 1986).

111. Rogers, H. Jr., *Theory of Recursive Functions and effective computability,* McGraw-Hill, New York (1967).

112. Rosenberg, A., "Nonwriting extensions of finite automata," Ph.D. Thesis, Harvard University (1965).

113. Rosenberg, A., "On multihead finite automata," *IBM J. Res. Develop.* **10**, pp. 388-394 (1966).

114. Schnorr, C.P., "Eine Bemerkung zum Begriff der zufällige Folge," *Z. Wahrsch. und verw. Geb.* **14**, pp. 27-35 (1969-1970).

115. Schnorr, C.P., "Zufälligkeit und Wahrscheinlichkeit; Eine algorithmische Begründung der Wahrscheinlichkeitstheorie," in *Lecture Notes in Mathematics*, Springer Verlag, Berlin (1971).

116. Schnorr, C.P., "Process complexity and effective random tests," *J. Comput. System Sciences* **7**, p. 376 (1973).

117. Schnorr, C.P., "A survey of the theory of random sequences," pp. 193-210 in *Basic Problems in Methodology and Linguistics*, ed. Butts, R.E., and J. Hintikka, D. Reidel, Dordrecht (1977).

118. Seiferas, J., "The symmetry of information, and An application of the symmetry of information," Notes, Computer Science Dept, University of Rochester (August 1985).

119. Shannon, C.E. and W. Weaver, *The Mathematical theory of Communication,* Univ. Illinois Press, Urbana, Ill. (1949).

120. Sipser, M., "A complexity theoretic approach to randomness," Proceedings 15th ACM Symposium on Theory of Computing (1983).

121. Solomonoff, R.J., "A preliminary report on a general theory of inductive inference," Tech. Rept. ZTB-138, Zator Company, Cambridge, Mass. (November 1960).

122. Solomonoff, R.J., "A formal theory of inductive inference, Part 1 and Part 2," *Information and Control* **7**, pp. 1-22, 224-254 (1964).

123. Solomonoff, R.J., "Complexity-based induction systems: comparisons and convergence theorems," *IEEE Transactions on Information Theory* **IT-24**, pp. 422-432 (1978).

124. Sudborough, I.H., "Computation by multi-head writing finite automata," Ph.D. Thesis, Pennsylvania State University, University Park (1974).

125. Sudborough, I.H., "One-way multihead writing finite automata," *Information and Control* **30**, pp. 1-20, (Also FOCS 1971) (1976).

126. Thompson, C.D., "Area-Time complexity for VLSI," 11th STOC (1979).

127. Turing, A.M., "On computable numbers with an application to the Entscheidungsproblem," *Proc. London Math. Soc.* **42**, pp. 230-265 (1936). Correction, *Ibid, 43* pp. 544-546 (1937).

128. Valiant, L. and G. Brebner, "Universal Schemes for parallel communication," 13th STOC (1981).

129. Valiant, L. G., "A Theory of the Learnable," *Comm. ACM* **27**, pp. 1134-1142 (1984).

130. Valiant, L. G., "Deductive Learning," *Phil. Trans. Royal Soc. Lond.* **A 312**, pp. 441-446 (1984).

131. Vazirani, U. and V. Vazirani, "A natural encoding scheme proved probabilistic polynomial complete," Proc. 23rd IEEE Symp. on Foundations of Computer Science (1982).

132. Ville, J., *Etude critique de la concept de Collectif,* Gauthier-Villars, Paris (1939).

133. Vitanyi, P.M.B. and L. Meertens, "Big Omega versus the wild functions," *SIGACT News* **16**(4), pp. 56-59 (1985).

134. Vitányi, P.M.B., "On the simulation of many storage heads by one," *Theoretical Computer Science* **34**, pp. 157-168 (1984).

135. Vitányi, P.M.B., "On two-tape real-time computation and queues," *Journal of Computer and System Sciences* **29**, pp. 303-311 (1984).

136. Vitányi, P.M.B., "Square time is optimal for the simulation of a pushdown store by an oblivious one-head tape unit," *Information Processing Letters* **21**, pp. 87-91 (1985).

137. Vitányi, P.M.B., "An N**1.618 lower bound on the time to simulate one queue or two pushdown stores by one tape," *Information Processing Letters* **21**, pp. 147-152 (1985).

138. Wald, A., "Sur la notion de collectif dans la calcul the probabilités," *Comptes Rendus des Sceánces de l'Acadeḿie des Sciences* **202**, pp. 1080-1083 (1936).

139. Watanabe, O., "Comparison of polynomial time completeness notions," *Theoretical Computer Science* **53** (1987).

140. Willis, D.G., "Computational complexity and probability constructions," *J. ACM* **17**, pp. 241-259 (1970).

141. Yao, A., "Theory and application of trapdoor functions," Proceedings 23rd IEEE Symposium on Foundations of Computer Science (1982).

142. Yao, A.C.-C. and R.L. Rivest, "k+1 heads are better than k," *J.ACM* **25**, pp. 337-340, (Also in Proc. 17th FOCS (1976), pp. 67-70.) (1978).

143. Zvonkin, A.K. and L.A. Levin, "The complexity of finite objects and the development of the concepts of information and randomness by means of the Theory of Algorithms," *Russ. Math. Surv.* **25**, pp. 83-124 (1970).