CHAPTER 4

Kolmogorov Complexity and its Applications

Ming LI

Aiken Computation Laboratory, Harvard University, Cambridge, MA 02138, USA

Paul M.B. VITÁNYI

Centrum voor Wiskunde en Informatica, Kruislaan 413, 1098 SJ Amsterdam, Netherlands, and Faculteit Wiskunde en Informatica, Universiteit van Amsterdam, Amsterdam, Netherlands

Contents

HANDBOOK OF THEORETICAL COMPUTER SCIENCE
Edited by J. van Leeuwen
© Elsevier Science Publishers B.V., 1990

1. Introduction

In everyday language we identify the information in an individual object with the essentials of a description for it. We can formalize this by defining the amount of information in a finite object (like a string) as the size (i.e., number of bits) of the smallest program that, starting with a blank memory, outputs 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 ones) contains little information because a program of size about $\log n$ outputs it (like "print n 1s"). Likewise, the transcendental number $\pi = 3.1415\ldots$, 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", "descriptive complexity", "program-size complexity", and others. Although there is a case to be made for "Solomonoff–Kolmogorov–Chaitin complexity" as the most appropriate name, we regard "Kolmogorov complexity" as well entrenched and commonly understood, and use it hereafter.

At the outset we wanted to survey the applications of this theory to the theory of computation, primarily in connection with the analysis and synthesis of algorithms in relation to the resources in time and space such algorithms require. But we were dealing with deep notions and general principles arising from, and having an impact to, many more disciplines. Gradually, the subject acquired a sophisticated mathematical theory and applications in an increasingly large number of astoundingly different areas. This chapter attempts to grasp the mass of fragmented knowledge of this fascinating theory.

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 purpose of this chapter 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. That is, we can use the fact that some strings are extremely compressible; that many strings are not compressible at all; and that some strings may be compressed but that it takes a lot of effort to 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 [144] 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 ones 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 ones.

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 to which 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 [71]. For instance, statistical survey of initial sequences of decimal digits of π have failed to disclose any significant deviations of randomness [71, 113, 147]. 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 [120]:

> "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 [115] proposed notions that approach the very essence of true randomness. In the early nineteenhundreds, Von Mises aimed at an axiomatic foundation of a calculus of probabilities. With the axioms validated by empirical evidence, in the manner of thermodynamics or mechanics, this would form a basis for real applications. However, while the ultimate justification of proposals for a proper theory of probabilities must lie in its applicability to real phenomena, this aspect was largely ignored in favor of the mathematical elegance of Kolmogorov's classic treatment of the set-theoretic axioms of his calculus of probability in 1933 [74].

> "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." [75].

Let us go into some more detail. The usual treatment of probability theory is 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 [115] Von Mises analyzes 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 $n \to \infty$. 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 by postulating that a sequence of outcomes of independent repetitions of random events in nature, like a sequence of tosses with a coin, satisfies certain properties. The properties selected were claimed to be validated by abundant empirical evidence. (We discuss the actual properties he suggests below.) The analogy Von Mises uses is with a physical science as thermodynamics, where, apart from the assumption of the basic laws like the law of conservation of energy, or the law of increasing entropy, the remainder is derived in a purely mathematical way. These laws have no other justification than a long history of failures of inventors of perpetuum mobiles (in contrast to the idea that perpetuum mobiles are impossible because of the laws of thermodynamics). Coming back to probability theory, granted the Von Mises axioms, the remainder of the calculus is developed in a purely mathematical way, and the mathematical laws of probability result. This approach actually satisfied Von Mises, and solves the problem noticed above because one property of a random sequence will be that the relative frequency limit exists. Other philosophers of science insist that additionally the random sequences defined form a set of full measure, and without exception do satisfy all laws of probability, because then it seems physically justifiable to assume that as a result of an (infinite) experiment only (or rather with probability one) random sequences appear (see [71, 100, 115, 173]).

Von Mises' particular interpretation of a notion of infinite random sequence of zeros and ones designated by the special name of *collective* (*Kollektiv* in German) is as follows. An infinite sequence $a_1 a_2 \ldots$ of zeros and ones is a random sequence in the special meaning of *collective* if the following two conditions are satisfied:

(1) Firstly, if f_n is the number of ones among the first n terms of the sequence, then

$$\lim_{n\to\infty} \frac{f_n}{n} = p,$$

for some p, $0 < p < 1$.

(2) Secondly, (1) is not only required for the original sequence, but (with the same limit p) also for every infinite subsequence $a_{n_1} a_{n_2} \ldots$ obtained by some *admissible* partial function ϕ, which is defined for all finite binary sequences and takes the values 0 and 1, and selecting one after the other those indices n for which $\phi(a_1 a_2 \ldots a_{n-1}) = 1$.

The existence of a relative frequency limit, condition (1), is a strong assumption. Empirical evidence from long runs of dice throws, in gambling houses, or with death statistics in insurance mathematics, suggests that the relative frequencies are *apparently convergent*. But clearly, no empirical evidence can be given for the existence of a definite limit for the relative frequency. However long the test run, in practice it will always be finite, and whatever the apparent behavior in the observed initial segment of the run, it is always possible that the relative frequencies keep oscillating forever if we continue.

Condition (2) says that, for any "admissible" strategy of successively selecting infinitely many elements from the sequence, the frequency of ones in the selection goes to the same limit as in condition (1). Put in other words, considering the sequence as fair coin tosses, condition (2) says there is no *strategy* ϕ (*principle of excluded gambling system*) that assures a player, betting at fixed odds and in fixed amounts on the tosses of the coin, to make infinite gain. That is, no advantage is gained in the long run by following some system, such as betting "head" after each run of seven consecutive tails, or (more plausibly) by placing the nth bet "head" after the appearance of $n+7$ tails in succession. According to Von Mises, the above conditions are sufficiently familiar and form an uncontroverted empirical generalization to serve as a basis of an applicable calculus of probabilities. 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, has profit in the long run from betting according to a system instead of betting at random. Says Church: "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 naive mathematical approach to a concrete formulation comes to grief as follows. We completely ignore the clear intention of Von Mises concerning a nontrivial restriction implied by the phrase "admissible place selection functions" by admitting simply *all* partial functions. Since arbitrary functions are allowed as a strategy, this definition is too restrictive, and no sequence exists that satisfies it with probability p other than 0.

EXAMPLE. Let $a = a_1 a_2 \ldots$ be any infinite string satisfying (1). Define ϕ_1 as $\phi_1(a_1 \ldots a_{i-1}) = 1$ if $a_i = 1$, and undefined otherwise. But then $p = 1$. However, this is not all of the story. Defining ϕ_0 by $\phi_0(a_1 \ldots a_{i-1}) = b_i$, b_i the complement of a_i for all i, we obtain by (2) that $p = 0$. Consequently, if we allow functions like ϕ_1 and ϕ_0 as strategies, then Von Mises' definition cannot be satisfied at all.

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 $\phi(a_1 \ldots a_{i-1})$. Here Von Mises' position is succinctly expressed by "first the collective, then the probability." Each collective, a physical object, determines what the admissible place selection functions for it are. While we can generalize from experience that all lawlike place selection functions are admissible, a place selection function pulled out of the blue with reference to a particular collective is inadmissible for it (but may be admissible for other collectives). In the example it just happens that *after* a criterion for admissible ϕ has been fixed too widely, 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. Here we cannot 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 [167] showed that the restriction of the set of admissible ϕ to a countable set eliminates the contradiction above. For any countable set of admissible selection functions, almost all sequences are random. Can we meaningfully fix some countable set of functions? A. Church proposed to fix it [37] to the formal notion of *effectively computable functions*, or *recursive functions*, as developed by A.M. Turing and himself (gamblers use computable strategies). He points out that, with a total recursive ϕ, not only is the definition completely rigorous and do corresponding random sequences exist, but moreover they are abundant since the infinite random sequences with $p = \frac{1}{2}$ form a set of measure 1; and from the existence of random sequences with probability $\frac{1}{2}$, the existence of random sequences associated with other probabilities is readily derived. Let us call sequences satisfying (1) and (2) with computable ϕ *Mises–Wald–Church* random. Appealing to a theorem by Wald yields as a corollary that the set of Mises–Wald–Church random sequences associated with any fixed probability has the cardinality of the continuum. Moreover, each Mises–Wald–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 Mises–Wald–Church random. This follows, for instance, from Champernowne's number

$$0.1234567891011121314151617181920\ldots$$

that is normal and where the ith digit is easily calculated from i. The definition of a Mises–Wald–Church random sequence implies that its consecutive digits cannot be effectively computed. (Namely, existence of an effective ϕ_1 as above contradicts $0<p<1$ in (2).) Thus, an existence proof for Mises–Wald–Church random sequences is necessarily nonconstructive.

Unfortunately, the Von Mises–Wald–Church definition is not yet good enough, since it was discovered by Ville [160] that even standard properties such as the Law of the Iterated Logarithm do not follow from it. In 1965, P. Martin-Löf, visiting Kolmogorov, succeeded in defining random sequences in a manner that is free of such difficulties [109]. His notion of infinite random sequences is related to infinite sequences of which all finite initial segments have high Kolmogorov complexity (cf. Section 2.4; for a survey of the work on infinite random sequences, see [81, 82]).

Until 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," [75].

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 [129]. However, properly defining random *finite* strings appeared to be an even more hopeless affair than such a definition for infinite strings. But the essence of the solution had already been discovered before. For instance, P.S. Laplace [83] and also Kolmogorov [75] observed 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

We feel it is important to give a careful treatment of the genesis of the ideas in this area. 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 [115] as discussed above.

With the advent of electronic computers in the 1950s, 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: R.J. Solomonoff in Cambridge, MA

[149], close in time but far away in geography followed by A.N. Kolmogorov in Moscow [75, 76], and then G.J. Chaitin in New York [25].

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 [22]. Already in November 1960 Solomonoff had published a Zator Company technical report on the subject of "Kolmogorov" complexity [148]. In March 1964 he published a long paper [149] introducing a version of universal a priori probability, namely the forerunner of the Solomonoff–Levin distribution, through the intermediate definition of what we have termed "Kolmogorov complexity", and proved the Invariance Theorem. This paper received little attention until Kolmogorov started to refer to it from 1968 onward. It is interesting to note that Solomonoff also discusses informally the ideas of randomness of finite strings, noncomputability of Kolmogorov complexity, computability of approximations to Kolmogorov complexity, and resource-bounded Kolmogorov complexity. A paragraph referring to Solomonoff's work occurs in [114]. To our knowledge, these are evidently the earliest documents outlining an algorithmic theory of descriptions.

In 1933 the great Soviet mathematician A.N. Kolmogorov[1] supplied probability theory with a powerful mathematical foundation [74]. 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 [76]. Kolmogorov's invention of the complexity of description was in no way a haphazard occurrence, but on the contrary the inevitable confluence of several of his major research threads: the foundations of probability and random sequences, information theory, and the theory of algorithms. *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 Solomonoff's work, in 1963–1964" [77].

G.J. Chaitin had finished the Bronx High School of Science, and was an 18-years old undergraduate student at the City College of the City University of New York when he submitted the original versions of [24, 25] for publication, in October and November 1965 respectively. Published in 1966, [24] investigated "state/symbol" complexities relative to arbitrary algorithms. In this work Chaitin extended C.E. Shannon's earlier work on coding concepts [145], and did not introduce any invariant notion of complexity. However, at the end of his 1969 publication [25], Chaitin apparently independently puts forward the proper notion of Kolmogorov complexity, proves the Invariance Theorem, and studies 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 about 1965 by A.N. Kolmogorov and me ... Both Kolmogorov and I were then unaware of related proposals made in 1960 by Ray Solomonoff" [27].

[1]Andrei N. Kolmogorov, born 25 April 1903 in Tambov, USSR, died 20 October 1987 in Moscow. For biographical details see [3, 17, 52], or [166], and the obituary in the *Times* [121].

The Swedish mathematician P. Martin-Löf, visiting Kolmogorov in Moscow during 1964–1965, investigated complexity oscillations of infinite sequences and proposed a new definition of infinite random sequences which is based on constructive measure theory [109, 111]. L.A. Levin, then a student of Kolmogorov, found a definition of a priori probability (the Solomonoff–Levin distribution) as a maximal semicomputable measure [173], and introduced the quantity corresponding to the self-delimiting variant of Kolmogorov complexity as the negative logarithm of a priori probability. In 1974 he more explicitly introduced Kolmogorov complexity based on self-delimiting programs [86]. This work relates to P. Gács' results concerning the differences between symmetric and asymmetric expressions for information [45]. In 1975 Chaitin also discovered and investigated this Kolmogorov complexity based on self-delimiting programs [28]. 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 [103, 104] and used extensively in [173]. Other variants and results were given by Willis [169], Levin [85], Schnorr [140], and Cover [38]. Apart from Martin-Löf's work, we mention that of C.P. Schnorr [138, 139] on the relation between Kolmogorov complexity and randomness of infinite sequences. This chapter of the Handbook is a precursor of our forthcoming textbook [97].

2. Mathematical theory

Kolmogorov gives a cursory but fundamental and elegant exposition of the basic ideas in [78]. Currently, the most complete treatment of the fundamental notions and results in Kolmogorov complexity is Levin and Zvonkin's 1970 survey [173]. Since this survey is not up to date, it should be complemented by Schnorr's [139, 140] and Chaitin's more recent monograph [30]. Kolmogorov and Uspenskii present a survey covering research in the Soviet Union [80], and Kolmogorov's selected works in the area are contained in [79] (see also [174]). For the advanced reader we mention Levin's important work [88]. An introductory but complete treatment of Kolmogorov complexity and its applications will be given in [97].

There are several variants of Kolmogorov complexity; here we focus on the original version in [24, 76, 78, 173]. (Later, we also define the more refined "self-delimiting" version.)

Notation. It is useful to fix some notation first. All through this paper it is convenient to identify the positive integers with the finite binary strings as follows:

$$(0, \varepsilon), (1, 0), (2, 1), (3, 00), (4, 01), (5, 10), (6, 11), (7, 000), \ldots$$

That is, the natural number n corresponds with the nth binary string in lexicographic length-increasing order. We call such an n a *finite object*, and whether we view it as a natural number or a binary string will be apparent from the context. If x is a binary string (natural number) then $|x|$ denotes the *length* or number of zeros and ones in x. The *number of elements* in a finite set A is denoted by $d(A)$. Hence, with $A = \{1, 2, \ldots, n\}$ we have $|d(A)|$ is about $\log n$. A few times we need to denote the *absolute value* of a number as in $|a-b|$, the absolute value of the difference of a and b. We feel that in each such case the context clearly indicates that we mean an absolute value and not a length, and refrain from introducing special notation.

First take a general viewpoint, as in [78], 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 a program the remaining bits of p. We say that a method S "absorbs a method S' with precision up to c" if for all x

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

Above we have shown how to construct a method S that absorbs 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 c-absorbs the other. As Kolmogorov remarks, this construction would be fruitless if the hierarchy of methods with respect to absorption were odd, for instance, if there is no bottom element. However, under relatively natural restriction on S this is not so. Namely, among the partial recursive functions (in the sense of Turing [155]), there exist *optimal* ones, say S, such that for any other computable function S'

$$K_S(x)\leqslant K_{S'}(x)+c_{S,S'}.$$

Clearly, all optimal methods S, S' of specifying objects in D are equivalent in the following way: the absolute value of the difference satisfies

$$|K_S(x)-K_{S'}(x)|\leqslant c_{S,S'}.$$

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 object x. 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 $p \in \{0,1\}^*$, and we call p a *program*. Any computable function f together with strings p and y such that $f(p,y)=x$ is a description of x. We call f the *interpreter* or *decoding* function. The (descriptional) complexity K_f of x, with respect to f, conditional to y, is defined by

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

and $K_f(x|y)=\infty$ if there are no such p. The following theorem asserts that each finite object has an intrinsic complexity which is independent from the means of description. Namely, there exist asymptotically optimal functions such that the description length with respect to them minorizes the description length with respect to any other function, apart from an additive constant, for all finite objects. This important fact is what makes the theory work.

Invariance Theorem (Solomonoff [149], Kolmogorov [76], Chaitin [25]). *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, $K_{f_0}(x|y)\leqslant K_f(x|y)+c_f$.*

Proof. Fix some standard enumeration of Turing machines, with an ordinary input tape, an extra input tape to contain the conditional information, a worktape, and an output tape. Let $n(T)$ be the number associated with Turing machine T. Assume that the conditional input y is contained on the extra input tape. Let f_0 be the *universal* partial recursive function computed by a universal Turing machine U. That is, U starting with input 0^n1p, $p\in\{0,1\}^*$, on the ordinary input tape and y on the extra input tape halts with output x on the output tape iff T starting with input p on the ordinary input tape and y on the extra input tape halts with x on its output tape, for $n(T)=n$. Choosing $c_f=n+1$ finishes the proof. □

Clearly, any function f_0 that satisfies the Invariance Theorem is *optimal* in the sense discussed above. Therefore, we are justified to fix a particular *reference machine U* as in the proof of the theorem and its associated partial recursive function f_0, and drop the subscripts on K. We define the *conditional Kolmogorov* complexity $K(x|y)$ of x under condition of y to be equal to $K_{f_0}(x|y)$ for this fixed optimal f_0. Define the *unconditional Kolmogorov* complexity of x as $K(x)=K(x|\varepsilon)$, where ε denotes the empty string ($|\varepsilon|=0$).

In his talks Kolmogorov used to credit A.M. Turing [155] for the universal Turing machine, which is the substance of the Invariance Theorem. 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 $|f(x)|\leqslant c\,|g(x)|$ for all $x\geqslant x_0$;
(ii) $f(x)=o(g(x))$ if $\lim_{x\to\infty} f(x)/g(x)=0$;
(iii) $f(x)=\Omega(g(x))$ if $f(x)\neq o(g(x))$; and
(iv) $f(x)=\Theta(g(x))$ if both $f(x)=O(g(x))$ and $f(x)=\Omega(g(x))$.

The relevant properties are extensively discussed in [72, 161]. 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. (Some computer scientists use the order of magnitude symbol Ω such that $f(x)=\Omega(g(x))$ iff there is a positive constant c such that $f(x)\geqslant cg(x)$ from some x onwards. This is different from our use of the traditional definition of Ω as the complement of o, as in (ii), which says: $f(x)=\Omega(g(x))$ iff there is a positive constant c such that $f(x)\geqslant cg(x)$ for infinitely many x.)

EXAMPLE. For each finite binary string x we have $K(xx)\leqslant K(x)+O(1)$. (The constant implied by the big-O notation is fixed by the choice of reference machine U.) 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 starting on $0^{n(T)}1p$ computes xx, and therefore U starting on $0^{n(V)}10^{n(T)}1p$ computes xx. Hence, for all x, $K(xx)\leqslant K(x)+n(V)+1$.

EXAMPLE. Let us define $K(x,y)=K(\langle x,y\rangle)$ with $\langle\cdot,\cdot\rangle$ a standard one-one mapping (pairing function) of pairs of natural numbers to natural numbers. That is, $K(x,y)$ is the length of a shortest program that outputs x and y and a way to tell them apart. It is seductive to conjecture $K(x,y)\leqslant K(x)+K(y)+O(1)$, 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 Section 2.2 on self-delimiting strings). Consequently, we have at best established

$$K(x,y)\leqslant K(x)+K(y)+O(\log(\min(K(x),K(y)))).$$

In general this cannot be improved.

2.1. Incompressibility

Apart from showing that complexity is an attribute of the finite object alone, the Invariance Theorem has also another most important consequence: it gives an upper bound on the complexity. Namely, there is a fixed constant c such that for all x of length n we have

$$K(x)\leqslant n+c.$$

This is easy to see. If T is a machine that just copies its input to its output, then $p=0^{n(T)}1x$ is a program for the reference machine U to output x.

This says that $K(x)$ is bounded above by the length of x modulo an additive constant. The obvious question to ask further is: "how many x can be compressed how far?". Since there are 2^n binary strings of length n, but only 2^n-1 possible shorter descriptions, it follows that, for all n, there is a binary string x of length n such that $K(x)\geqslant n$. 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 $K(x|y) \geqslant n$.

Example. Is a substring of an incompressible string also incompressible? A string $x = uvw$ can be specified by a short description for v of length $K(v)$, a description of $|u|$, and the literal description of uw. Moreover, we need information to tell these three items apart. Such information can be provided by prefixing each item with a self-delimiting description of its length, as explained in the section on self-delimitation. Together this takes $K(v) + |uw| + O(\log|x|)$ bits. Hence,

$$K(x) \leqslant K(v) + O(\log|x|) + |uw|.$$

Thus, if we choose x incompressible, $K(x) \geqslant |x|$, then we obtain

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

It can be shown that this is optimal—a substring of an incompressible string can be compressible. This conforms to a fact we know from probability theory: every sufficiently long random string must contain long runs of zeros.

Example. Define $p(x)$ as a shortest program for x. We show that $p(x)$ is incompressible, in the sense that there is a constant $c > 0$ such that for all strings x, we have $K(p(x)) \geqslant |p(x)| - c$. Suppose the contrary. 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 $K(x) \leqslant |p(x)| - c + n(V) + 1$, for all $c > 0$, some x, which is impossible.

Example. It is easy to see that $K(x|x) \leqslant n(T) + 1$, where T is a machine that just copies the input to the output. However, it is more interesting that, for *some* shortest program $p(x)$ of (x), $K(p(x)|x) \leqslant \log K(x) + O(1)$, which cannot be improved in general. Hint: later we show that K is a noncomputable function. This rules out that we can compute any shortest program $p(x)$ from x. However, we can dovetail the computation of all programs shorter than $|x| + 1$: run the first program one step, run the first program one step and the second program one 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 any such 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, there is at least one x of length n that cannot be compressed to length $< n$ since there are 2^n strings of length n and but $2^n - 1$ programs of length less than n; at least $\frac{1}{2}$ of all strings of length n cannot be compressed to length $< n-1$ since there are but $2^{n-1} - 1$ programs of length less than $n-1$; at least $\frac{3}{4}$th of all strings of length n cannot be compressed to length $< n-2$, and so on.

Generally, let $g(n)$ be an integer function. Call a string x of length n *g-incompressible* if

$K(x) \geqslant n-g(n)$. There are 2^n binary strings of length n, and only $2^{n-g(n)}-1$ possible descriptions shorter than $n-g(n)$. Thus, the ratio between the number of strings x of length n with $K(x)<n-g(n)$ and the total number of strings of length n is at most $2^{-g(n)}$, a *vanishing fraction* when $g(n)$ increases unboundedly with n. In general we loosely call a finite string x of length n *random* if $K(x) \geqslant n-\mathrm{O}(\log n)$.

Intuitively, incompressibility implies the absence of regularities, since regularities can be used to compress descriptions. Accordingly, we like to identify incompressibility with absence of regularities or *randomness*. In the context of finite strings randomness like incompressibility is a matter of degree: it is obviously absurd to call a given string random and call nonrandom the string resulting from changing a bit in the string to its opposite value. Thus, we identify c-incompressible strings with *c-random* strings.

However, with infinite strings we may a priori hope to be able to use Kolmogorov complexity to sharply distinguish the random strings from the nonrandom ones, to finish the task set by Von Mises (see the Introduction). Let us call an infinite string x *g-incompressible* if each initial string $x_{1:n}$ of length n has $K(x_{1:n}) \geqslant n-g(n)$, from some n onward. In [109], Martin-Löf has defined a satisfactory notion for randomness of infinite strings. It turns out that Martin-Löf random strings are $(2\log n)$-incompressible, but not $(\log n)$-incompressible (cf. Section 2.4). We call finite or infinite $\mathrm{O}(\log n)$-incompressible strings loosely "random" or "Kolmogorov random", but want to stress here that randomness for infinite strings according to Martin-Löf has a stricter and 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 an information-theoretic version of Gödel's Theorem below. Strings that are not incompressible are *compressible* or *nonrandom*. The nonrandom infinite binary strings are very scarce: they have measure zero in the set of all infinite binary strings.

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 $x, y, x_i, y_i, \ldots$ 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 $\mathrm{O}(1)$ 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 C.E. Shannon.

For each string $x \in \{0,1\}^*$, the string $\bar{x}$ is obtained by inserting a "0" in between each pair of adjacent letters in x, and adding a "1" at the end. That is,

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

Let $x'=\overline{|x|}\,x$ (an encoding of the length of x in binary followed by x in binary). The string x' is called the *self-delimiting* version of x. So "100101011" is the self-delimiting version of "01011". (According to our convention "10" is the fifth binary string.) 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 $|n|$ of a natural number n by "$\log n$".

Example (*generalization*). More generally, for $x\in\{0,1\}^*-\{\varepsilon\}$, $d_0(x)=\bar{x}$ is the self-delimiting version of order 0 of x using $2|x|$ bits. Above we defined the "standard" self-delimiting version $d_1(x)=x'$ of order 1. In general, for $i\geqslant 1$, $d_i(x)=\bar{x}_i x_{i-1}\ldots x_1 x$, with $x_1=|x|$ and $x_j=|x_{j-1}|$ $(1<j\leqslant i)$, is the self-delimiting version of order i of x. Define $\log^{(1)}=\log$, and $\log^{(j+1)}=\log\log^{(j)}$ for $j\geqslant 1$. Then,

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

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 $K_c(x)=\min\{|c(M)y|: M$ on input y 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 define a variant of Kolmogorov complexity by requiring at the outset that we only consider Turing machines for which the set of programs is a prefix code. The resulting variant, called *self-delimiting* Kolmogorov complexity, has nicer mathematical properties than the original one, and has therefore become something of a standard in the field. This complexity is variously denoted in the literature by KP, I, H, or simply by K which results in confusion with the original notion. We treat it in Section 2.7 and denote it (only) there by K'. For most applications it does not matter whether we use $K'(x)$ or $K(x)$ since they coincide to within an additive term of $O(\log|x|)$. In the case of inductive inference, however, we need to use the self-delimiting version of complexity. We denote both versions indiscriminately by $K(x)$, and point out which version we mean if it matters.

Example (*Li, Maass, and Vitányi*). 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 $x_1\ldots x_k$ be a binary string of length n with the x_i's

$(1 \leqslant i \leqslant k)$ 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 ($m \leqslant d$) is the number of "holes" in the string, and the literal representation of $\hat{x} = \hat{x}_1 \hat{x}_2 \ldots \hat{x}_k$.

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 (nonempty) $\hat{x}_i$'s, with

$$k = \sum_{i=1}^{m} (p_i + d_i) \quad \text{and} \quad d = \sum_{i=1}^{m} d_i.$$

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

$$(k-d)c + 3d \log(k/d) + \mathrm{O}(\log n).$$

We then proceed by showing that we can describe x by this description plus some description of the deleted x_i's, so that the total requires considerably less than n bits. Choosing x such that $K(x) \geqslant n$ then gives a contradiction (see [94]).

This finishes the Application Toolkit. We have now formalized the essence of what we need for most applications in the sequel. Having made our notions precise, many applications can be described informally yet rigorously. The remainder of the theory of Kolmogorov complexity we treat below is not always required for the later applications. But, for instance, for the proper definition of the Solomonoff–Levin distribution, as needed in the application to inductive inference, it is required to use the self-delimiting version of complexity we briefly discussed in an example above (see also Section 2.7).

2.3. *Quantitative estimate of K*

We want to get some insight in the quantitative behavior of K. We follow [173]. 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 $d(M)$ denote the number of elements in M. Then the fraction of $x \in M$ for which $K(x|y) < |d(M)| - m$, does not exceed 2^{-m}, by a counting argument similar to that in Section 2.1. 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 cannot be significantly more either.

LEMMA (Kolmogorov). *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, $K(x|y) \leqslant |d(M_y)|$.*

PROOF. Let A be enumerated by a Turing machine T. Using y, modify T to T_y such that T_y enumerates all pairs (x, y) in A, without repetition. In order of enumeration we select the pth pair (x, y), and output the first element, i.e. x. Then we find $p < d(M_y)$, such that $T_y(p) = x$. Therefore, we have by the Invariance Theorem $K(x|y) \leqslant K_{T_y}(x) \leqslant |d(M_y)|$, as required. □

Example. Let A be a subset of $\{0,1\}^*$. Let $A^{\leqslant n}$ equal $\{x \in A: |x| \leqslant n\}$. If the limit of $d(A^{\leqslant n})/2^n$ goes to zero for n going to infinity, then we call A *sparse*. For example, the set of all finite strings that have twice as many zeros as ones is sparse. This has as a consequence that all but finitely many of these strings have short programs.

Claim. (a) (Sipser) *If A is recursive and sparse, then for all constant c there are only finitely many x in A with $K(x) \geqslant |x| - c$. Using Kolmogorov's Lemma we can extend this result as follows.*

(b) *If A is r.e. and $d(A^{\leqslant n})/n^{-(1+\varepsilon)}2^n$, $\varepsilon > 0$, goes to zero for n going to infinity, then, for all constant c, there are only finitely many x in A with $K(x) \geqslant |x| - c$.*

(c) *If A is r.e. and $d(A^{\leqslant n}) \leqslant p(n)$ with p a polynomial, then, for all constant $c > 0$, there are only finitely many x in A with $K(x) \geqslant |x|/c$.*

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 $K(x) \leqslant K(i) + d$. If x has length n, then the sparseness of A implies that $K(i) \leqslant n - g(n)$, with $g(n)$ unbounded. 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

$$K(x|n) \leqslant n - (1+\varepsilon)\log n + O(1).$$

Therefore, $K(x) \leqslant n - \varepsilon \log n + O(1)$, for some other fixed positive ε, and the right-hand side of the inequality is less than $n - c$ from some n onward.

(c) Similarly as above. □

We now look at unconditional complexity. We identify the binary string x with the natural number x, as in the correspondence mentioned at the outset of Section 2. This way, $K(x)$ can be considered as an integer function.

Lemma (Kolmogorov). *For any binary string x, the following hold:*

(a) *$K(x) \leqslant |x|$, 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}, so that equality holds in* (a) *for the majority of words.*

(c) $\lim_{x \to \infty} K(x) = \infty$, *and*

(d) *for $m(x)$ being the largest monotonic increasing integer function bounding $K(x)$ from below, $m(x) = \min_{y \geqslant x} K(y)$, 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 $m(x) < \phi(x)$, for all large enough x.*

(f) *The absolute difference $|K(x+h) - K(x)| \leqslant 2|h|$, up to some constant independent of x, h. That is, 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 [173]. 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 $|\bar{h}|\leqslant 2|h|$ (see previous section), this proves (f). □

EXAMPLE. One effect of the information quantity associated with the length of strings is that $K(x)$ is *nonmonotonic on prefixes*. This can be due to the information contained in the length of x. That is, clearly for $m<n$ we can still have $K(m)>K(n)$. But then $K(x)<K(y)$ for $x=0^n$ and $y=0^m$, notwithstanding that y is a proper prefix of x. For example, if $n=2^k$ then $K(0^n)\leqslant \log\log n+O(1)$, while we have shown above that there are $m<n$ for which $K(m)\geqslant \log n-O(1)$. Therefore, the complexity of a part can turn out to be greater than the complexity of the whole. In an initial attempt to solve this problem we may try to eliminate the effect of the length of the string on the complexity measure by treating the length as given. However, this does not help as the next example shows.

EXAMPLE. For any binary string x, $|x|=n$, we have $K(x|n)\leqslant K(x)$, but usually the length of x does not give too much information about x. But sometimes it does. For instance for a string $x=0^n$. Then, $K(x)=K(n)+O(1)$, but $K(x|n)=O(1)$. Moreover, it is easy to find m such that $m<n$ and $K(0^m|n)=\Omega(\log n)$. But of course $K(0^m|m)=O(1)$ again. Thus, our next try is to look at the complexity $K(x|n)$ where $n=|x|$. But now we get nonmonotonicity in another way. Consider $x=n00\ldots0$ with $|x|=n$, that is, the nth binary string padded with zeros up to length n. These strings are called *n-strings* by Loveland [104]. Now always $K(x|n)=O(1)$, but by choosing the prefix n of x random we have $K(n|m)=\Omega(\log n)$ with $m=|n|$.

2.4. *Infinite random strings*

If $x=x_1x_2\ldots$ is a finite or infinite string of binary digits x_i, $|x|\geqslant n$, then $x_{m:n}$ denotes the substring $x_mx_{m+1}\ldots x_n$.

In connection with the task set by Von Mises (cf. Introduction) we would like to express the notion of randomness of infinite binary strings in terms of Kolmogorov complexity. The obvious way is to use the definition of finite random strings. That is, call an infinite binary string x random if there is a constant c such that, for all n, $K(x_{1:n})\geqslant n-c$. But in 1965 Martin-Löf found that such strings do not exist [109–111]:

THEOREM (Martin-Löf). *If $f(n)$ is a recursive function such that $\Sigma 2^{-f(n)}=\infty$, then for any infinite binary sequence x there are infinitely many n for which $K(x_{1:n})<n-f(n)$.*

EXAMPLE. $f(n)=\log n$ satisfies the condition of the theorem. Let $x=x_1x_2\ldots$ be any infinite binary string, and $x_{1:m}$ any m-length prefix of x. If $n-m$ is the natural number corresponding to $x_{1:m}$, so m is about $\log n$, then $K(x_{1:n})=K(x_{m+1}\ldots x_n)+O(1)$. This is easy to see, since we can uniquely reconstruct $x_{1:m}$ from the length $n-m$ of $x_{m+1}\ldots x_n$ with $O(1)$ additional bits of information.

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

$$\sum 2^{-f(n)} \tag{2.1}$$

converges recursively (there is a recursive set of integers n_r such that $\sum_{n \geqslant n_r} 2^{-f(n)} \leqslant 2^{-r}$, for example $f(n) = \log n + 2 \log \log n$), then almost all strings x (in the sense of binary measure) have the property

$$K(x_{1:n}) \geqslant n - f(n), \tag{2.2}$$

from some n onwards. In a less precise form these phenomena were also presented by Chaitin [25]. Due to these complexity oscillations the idea of identifying infinite random sequences with those such that $K(x_{1:n}) \geqslant n - c$ does not work. These problems caused Martin-Löf to try another track and proceed directly to the heart of the matter. Namely, to justify any proposed definition of randomness, one will have to show that the sequences that are random in the stated sense satisfy the several properties of stochasticity we know from the theory of probability. So why not, instead of proving that each such property separately is satisfied by a proposed definition, formalize the property that the random sequences introduced possess, in an appropriate sense, all possible properties of stochasticity.

It turns out that the notion of infinite binary strings satisfying all properties of randomness, in the sense of all properties that hold with probability 1, is contradictory. However, if we restrict ourselves to only those properties that are effectively verifiable, and statistical tests for randomness invariably are effective, then the resulting notion of random infinite string is noncontradictory. Pursuing this approach through constructive measure theory, Martin-Löf [109] 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". That is to say, 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 $\frac{1}{2}$. Not only do such random strings exist, indeed, it turns out that these random strings have measure 1 in the set of all strings. Using this definition of randomness he shows the following theorems [111].

Theorem (Martin-Löf). *Random binary sequences satisfy* (2.2) *from some n onwards, provided* (2.1) *converges recursively.*

Theorem (Martin-Löf). *If $K(x_{1:n}|n) \geqslant n - c$ for some constant c and infinitely many n, then x is a random binary sequence.*

For related work see also [103, 138, 139, 141, 173]. We mention that for the self-delimiting version (Section 2.7) of complexity K' it holds that x is random iff there is a constant $c > 0$ such that $K'(x_{1:n}|n) \geqslant n - c$, for all n.

2.5. *Algorithmic properties of K*

We select some results from Zvonkin and Levin's survey [173]. Again, we consider $K(x)$ as a function that maps a positive integer x to a positive integer $K(x)$.

THEOREM (Kolmogorov). (a) *The function $K(x)$ is not partial recursive. Moreover, no partial recursive function $\varphi(x)$, 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 $\lim_{t\to\infty} H(t, x) = K(x)$. That is, we can obtain arbitrary good estimates for $K(x)$ (but not uniformly).*

PROOF. (a) Every infinite r.e. set contains an infinite recursive subset [135, Theorem 5-IV]. Select an infinite recursive set A in the domain of definition of $\varphi(x)$. The function $f(m) = \min\{x: K(x) \geqslant m, x \in A\}$ is (total) recursive (since $K(x) = \varphi(x)$ on A), and takes arbitrarily large values. Also, by construction, $K(f(m)) \geqslant m$. On the other hand, $K(f(m)) \leqslant K_f(f(m)) + c_f$ by definition of K, and obviously $K_f(f(m)) \leqslant |m|$. Hence, $m \leqslant \log m$ up to a constant independent of m, which is false.

(b) Let c be a constant such that $K(x) \leqslant |x| + c$ for all x. Define $H(t, x)$ as the length of the smallest program p, with $|p| \leqslant |x| + c$, such that the reference machine U with input p halts with output x within t steps. □

EXAMPLE (Barzdin'). It is not too difficult to show by similar reasoning that if $f(x) < |x|$ is a total recursive function with $\lim_{x\to\infty} f(x) = \infty$, then the set $B = \{x: K(x) \leqslant f(x)\}$ is *simple* in the recursive-theoretic sense of Post. That is, B is recursively enumerable and the complement of B is infinite but does not contain an infinite recursively enumerable subset. It is then straightforward that for every axiomatized theory F (that is consistent and sound) there are only finitely many n for which the statement "$n \notin B$" is both true and provable in F. However, from the definition of B it follows that all x with $K(x) \geqslant |x|$ do not belong to B, and there are infinitely many of those. Hence, if F is strong enough to express statements of the form "$n \notin B$", for instance F contains arithmetic, then infinitely many true statements can be expressed in F but are not provable. This is a version of Gödel's famous incompleteness result. It is different from Gödel's original proof in the fact that our undecidable statements are not constructive. This result is attributed to Barzdin' in Levin and Zvonkin's 1970 survey [173].

It turns out that with Kolmogorov complexity one can quantify the distinction between r.e. sets and recursive sets. Let $x = x_1 x_2 \ldots$ be an infinite binary sequence such that the set of numbers n with $x_n = 1$ is r.e. That is, x is the *characteristic sequence* of the set $M = \{n: x_n = 1\}$. 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 $K(x_{1:n}|n)$ bounded. But in the general case, when the set of ones is r.e., $K(x_{1:n})$ can grow unboundedly.

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

PROOF. Let the number of ones in $x_{1:n}$ be $m \leqslant n$. Since M is r.e. we can recursively enumerate all of its elements without repetition. Given m, we know that after having enumerated m elements in M that are less or equal to n, we have found them all. Since

$K(m) \leqslant \log n + c$ for some fixed constant c, this proves the upper bound. The lower bound holds for universal sets like $K_0 = \{\langle x, y\rangle : T_x \text{ halts on input } y\}$ (see [9, 103, 173]). □

In [78] Kolmogorov gives the following interesting interpretation with respect to investigations in the foundations of mathematics: Label all Diophantine equations by natural numbers. Y.V. Mateyasevich 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 Tenth 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 $\log n + O(1)$. 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. Section 3.6).

In the same 1968 paper [9] 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 and T be a Turing machine. Define $K_T^t(x_{1:n}|n)$ as the minimum length of a program p such that T starting with conditional n on its extra input tape computes the n-length prefix of x within $t(n)$ steps, and then halts.

THEOREM (Barzdin'). *Let T be any Turing machine. 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, $K_T^t(x_{1:n}|n) \leqslant cn$ holds. Moreover, there are such sequences x such that for any (total) recursive t and any n, $K_T^t(x_{1:n}) \geqslant c_t n$ holds, with c_t a constant independent of n (but dependent on t.)*

2.6. *Information*

If the conditional complexity $K(x|y)$ 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

$$I(x:y) = K(y) - K(y|x)$$

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

$$K(x|x) = 0, \qquad I(x:x) = K(x).$$

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 [144]. 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 (see the examples *before* Section 2.1). That is, 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* $\mathrm{O}(\log K(x,y))$,

$$K(x,y)=K(x)+K(y\,|\,x).$$

In the general case it has been proved that equality up to a logarithmic error term is the best possible. From the lemma it follows immediately that, to within an additive term of $\mathrm{O}(\log K(x,y))$,

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

and therefore the absolute value of the difference of the information quantities satisfies

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

It has been established that the difference can be of this order (see [173]).

2.7. *Self-delimiting Kolmogorov complexity*

This more refined version of complexity is, in a sense, implicit in Solomonoff's original a priori probability [148, 149]. A definition (the universal semicomputable semimeasure $\boldsymbol{m}(x)$, corresponding to the Solomonoff–Levin distribution we study later) was supplied in the 1970 survey of Levin and Zvonkin [173]. The quantity corresponding to the self-delimiting Kolmogorov complexity $K'(x)$ occurs already in the form of the negative logarithm of the a priori probability $\boldsymbol{m}(x)$. It is explicitly defined as below and studied by Levin and Gács in 1974 [45, 86]. It was also discovered in 1975 by Chaitin [28]. For the development of the theory $K'(x)$ is often a more useful complexity measure than the $K(x)$, but for many applications one can use both equally well because they coincide to within a logarithmic factor.

NOTE. With some abuse of notation, after this section we simply drop the prime of $K'(x)$ and denote all types of Kolmogorov complexity simply by $K(x)$. If it is important whether we intend the self-delimiting version or the non-self-delimiting one, then we will explicitly state which version we mean.

There are two ways to define $K'(x)$. First, consider a class of Turing machines $T_1, T_2, \ldots$ with a one-way input tape, a one-way output tape, and a two-way worktape. Let the infinite input tape contain only zeros or ones (no blanks). These are the self-delimiting Turing machines, because the set of inputs (programs) for which each machine halts is prefix-free. We call a binary string p a *program* for T if T starts scanning the leftmost bit of p and halts scanning the rightmost bit of p. Just as before, we can prove an Invariance Theorem concerning the complexities associated with the different machines, where the optimal complexity is provided by the universal self-delimiting machine. We fix one such machine U and call it the reference self-delimiting machine. The *self-delimiting complexity* $K'(x)$ is the length of the shortest program p of U that outputs x.

FACT. We have defined programs so that no program is the prefix of another one. Each program is *self-delimiting* with respect to T. This allows an alternative approach to define $K'(x)$.

Second approach: For a self-delimiting machine T and each binary string x, define $P(x)$ as the *probability* that T eventually halts with x written on the output tape. (Solomonoff has called P the a priori probability, cf. Section 2.9.) The *entropy* is defined as $H(x) = -\log P(x)$.

Let the symbols on the input tape be provided by independent tosses of an unbiased coin. This enables us to give a natural probability distribution over programs: the probability of program p is simply $2^{-|p|}$. We can now 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 x by a program of length $|x|+2\log|x|$. Namely, define Turing machine T such that it outputs a binary string x iff it first reads the self-delimiting binary encoding of the length of x, and then the usual binary representation of x. Thus, with respect to T,

$$P(x) \geqslant 2^{-|x|-2\log|x|}, \qquad H(x) \leqslant |x|+2\log|x|, \qquad K'(x) \leqslant |x|+2\log|x|.$$

To make the definitions meaningful, we normalize these measures with respect to an optimal *universal* machine. This choice maximizes P and minimizes H and K'. It can be shown that the optimal self-delimiting machine selected this way can be set equal to the reference machine U of the earlier approach! Hence the two approaches define the same K' (as usual, up to a fixed additive constant). Let U be such a machine. Then, the a priori probability of x is

$$P(x) = \sum_{U(p)=x} 2^{-|p|}.$$

(This is the *Solomonoff–Levin* distribution which we denote below by the special notation $\boldsymbol{m}(x)$. Thus the entropy $H(x) = -\log \boldsymbol{m}(x)$.) It follows immediately from the definitions that the relation between the different notions is $K(x) \leqslant K'(x) \leqslant H(x)$. Levin has shown the significant result that $K'(x) = -\log \boldsymbol{m}(x) + O(1)$. That is, if x has many *long* programs, it must also have a *short* program.

EXAMPLE. To within an additive constant, for all finite binary strings x, y we have $K'(x,y) \leqslant K'(x)+K'(y)$. Namely, let p and q be self-delimiting 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 x, y. Presented with input pq, V can tell p apart from q because p is self-delimiting. Hence, U with program $0^{n(V)}1pq$, computes xy. Therefore, for all finite binary strings x, y, $K'(x, y) \leqslant K'(x) + K'(y) + n(V) + 1$.

The self-delimiting complexity K' satisfies many laws without a logarithmic fudge term. Infinite random sequences can be more naturally defined using K' complexity. In fact, it turns out that an infinite binary string x is random in the sense of Martin-Löf (see before and [109]) iff there exists a constant c such that $K'(x_{1:n}) \geqslant n - c$ for all n. This interesting characterization was proposed by Chaitin [28], and proved by Schnorr. The following lemma, due to Levin and Gács [45], and also Chaitin [28], shows that $K'(x)$ is a symmetric measure of the information in x.

LEMMA (strong symmetry). *To within an additive constant,*

$$K'(x, K'(x)) = K'(x), \qquad K'(x, y) = K'(x) + K'(y \mid (x, K'(x))).$$

Therefore, the exact symmetry of information holds up to an additive constant in the sense that

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

REMARK. It is a fundamental result due to Gács [45] that if we delete $K'(x)$ and $K'(y)$ from the condition, the equalities in the lemma in general can only hold to within a term logarithmic in $K'(x, y)$. In Chaitin's formulation [28] of conditional complexity, say denoted by $Kc(x \mid y)$, he means the complexity of x given the lexicographically least program p for y. It is straightforward to verify that $Kc(x \mid y) = K'(x \mid p) = K'(x \mid (y, K'(y)))$. Furthermore, if simply $Kc(x) = K'(x)$ and $Kc(x, y) = K'(x, y)$, then we have $Kc(x, y) = Kc(x) + Kc(y \mid x)$ up to an additive constant.

2.8. Probability theory

Laplace [83] 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, one-hundred heads are just as probable as any other equal-length sequence of heads and tails, even though we feel that it is less "random" than some others. We can formalize this. (We follow Gács's insightful treatment [46] in this section). Let us call a *pay-off* function (or *martingale*) with respect to distribution P any nonnegative function $t(x)$ with $\sum_x P(x)t(x) \leqslant 1$. 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 ought to agree to pay

$t(x)$ dollars on outcome x. Accordingly, we propose a pay-off 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 pay-off 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 by Levin. We need the *Kraft inequality*: for each set S of finite binary strings such that no string in S is a proper prefix of another string in S, we have

$$\sum_{x \in S} 2^{-|x|} \leqslant 1.$$

We call S a *prefix-code*. With notation $K(x|y)$ we mean the self-delimiting complexity of the previous section. Then for each fixed y, the set of $K(x|y)$'s is the length set of a prefix-code, so Kraft's inequality applies. 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. By Kraft's inequality, for each fixed y,

$$\sum_{x} 2^{-K(x|y)} \leqslant 1, \tag{2.3}$$

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

$$K(x) \leqslant -\log \mu(x) + K(\mu) + c, \tag{2.4}$$

with c a universal constant. Put $d(x|\mu) = -\log \mu(x) - K(x)$. By (2.3), $t(x|\mu) = 2^{d(x|\mu)}$ is a pay-off function. We can now beat any fraudulent casino. We propose the pay-off function $t(x) = 2^{-\log P_n(x) - K(x|n)}$. (We use conditional complexity $K(x|n)$ because the uniform distribution P_n depends on n.) If every other coin flip comes up heads, then $K(x|n) \leqslant (\frac{1}{2}n) + c_0$, and hence we win $2^{t(x)} \geqslant c_1 2^{n/2}$ from the casino ($c_0, c_1 > 0$), even though the bet does not refer to "heads".

The fact that $t(x|\mu)$ is a pay-off function, implies by Chebychev's First Inequality that for any $k > 0$,

$$\mu\{x : K(x) < -\log \mu(x) - k\} < 2^{-k}. \tag{2.5}$$

Together, (2.4) and (2.5) 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 pay-off function $t(x)$ taking large values on the outcomes violating the law. In general, the pay-off function $t(x|\mu)$ is *maximal* (up to a multiplicative constant) among all pay-off functions that are semicomputable (from below). Hence the quantity $d(x|\mu)$ 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 Solomonoff–Levin distribution*

Let $K(x)$ be the self-delimiting variant of complexity. The incomputable *Solomonoff–Levin distribution* $\boldsymbol{m}(x)$ can be defined as

$$\boldsymbol{m}(x) = \sum_{U(p)=x} 2^{-|p|},$$

with U the reference universal self-delimiting Turing machine. Now $\boldsymbol{m}(x)$ can be interpreted as the probability that U halts with output x if we generate the input p by an indefinitely long sequence of random coin flips (U will use only the self-delimiting prefix p of this sequence as its program) (see also the section on self-delimiting complexity). This distribution was conceived by Solomonoff in [148, 149], but in different form. A mathematically natural form, in terms of a *universal semicomputable semimeasure* that dominates all semicomputable semimeasures, which can be shown to coincide with the above definition of $\boldsymbol{m}(x)$ up to a multiplicative constant, was given by Levin in [173]. A discrete semimeasure μ is a real-valued function satisfying $\sum_x \mu(x) \leqslant 1$. A function μ is semicomputable (from below) if the set $\{\langle p,q,x\rangle\colon p/q \leqslant \mu(x)\}$, p,q and x natural numbers, is recursively enumerable.

It can be shown that $\boldsymbol{m}$ is a universal semicomputable semimeasure in the sense that, for each semicomputable semimeasure μ, there exists a constant $c>0$ such that, for all x, $\boldsymbol{m}(x) > c\mu(x)$. That is, $\boldsymbol{m}$ dominates μ multiplicatively.

It can be shown that $\boldsymbol{m}(x) = 2^{-K(x)\pm O(1)}$. It turns out that $\boldsymbol{m}(x)$ has the remarkable property that the test $d(x|\boldsymbol{m})$ shows all outcomes x random with respect to it. We can interpret (2.4) and (2.5) as saying that if the real distribution is μ, then $\mu(x)$ and $\boldsymbol{m}(x)$ are close to each other with large probability. Therefore, if x comes from some unknown computable distribution μ, then we can use $\boldsymbol{m}(x)$ as an estimate for $\mu(x)$. Accordingly, Solomonoff has called $\boldsymbol{m}$ "a priori probability". The randomness test $d(x|\mu)$ can be interpreted in the framework of hypothesis testing as the likelihood ratio between hypothesis μ and the fixed alternative hypothesis $\boldsymbol{m}$. 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 Section 3.2).

Since the a priori probability $\boldsymbol{m}$ 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 binary sequences as in [173]. 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

$$\frac{M(xy)}{M(x)} \tag{2.6}$$

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 [150]. Inductive inference formula (2.6) 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" [46].

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 $\pi = 3.1415\ldots$ and $e = 2.7182\ldots$. 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 $O(1)$. A nice application of what we may call extreme compressibility of some strings is a new version of Gödel's celebrated incompleteness theorem.

3.1. A version of Gödels Theorem

Recall 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. Barzdin' has first formulated a new form of Gödel's Incompleteness Theorem in terms of simple sets (cf. Section 2.5). This was also treated by Chaitin in a sequence of papers [26, 27, 32], as follows. Let us 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, the argument below shows that the worst-case such ratio expressed as a function of the length of the theorem increases faster than any computable function.

In Bennett's [11] phrase: *although most numbers are random, only finitely many of them can be proved random within a given consistent axiomatic system.* If T is an axiomatized system (which is sound, i.e. all theorems provable in T are true), whose axioms and rules of inference require about k bits to describe, then T 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, then 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. (This formulation is due to Bennett.)

EXAMPLE (*Chaitin*). We use an approach that is slightly different from Chaitin's approach.

(a) Let axiomatized theory T be describable in k bits: $K(T) \leqslant k$. Assume that T is contradiction-free.

(b) Assume that all true formulas in T can be proved in T.

(c) Let $S_c(x)$ be a formula in T with the meaning: "x is the lexicographically least binary string of length c with $K(x) \geqslant c$." Here x is a formal parameter and c an explicit constant, so $K(S_c) \leqslant \log c$ up to a fixed constant independent of T and c.

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 T, S_c, 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 $\text{not}(S_c(y)) = \mathbf{true}$ (holds for $y \neq x$), by simple enumeration of all proofs in T. We need to distinguish the descriptions of T and S_c, so we code T's description self-delimiting in not more than $2k$ bits. Hence, for some fixed constant c' independent of x, T and c, we find $K(x) \leqslant 2k + \log c + c'$, which contradicts $K(x) > c$ for all $c > c_T$, where $c_T = 3k + c'$ for another constant c'.

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."

EXAMPLE (*Levin*). Levin has tried to gauge the implications of algorithmic complexity arguments to probe depths beyond the scope of Gödel's arguments. He derives what he calls "Information Conservation Inequalities". Without going into the subtle details, Levin's argument in [86] 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 other terms, any sequence that may arise in nature contains only a finite amount of information about any sequence defined mathematically. Levin says: "Our thesis 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 [88].)

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 universal a priori probability, as described previously. Solomonoff's proposal [149], i.e., the Solomonoff–Levin distribution, 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 and 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, we give a simplified Solomonoff inductive inference theory. Given a previously observed data string S over $\{0,1\}^*$, the inference problem is to predict the next symbol in the output sequence, i.e., extrapolating the sequence S. By Bayes' rule, for $a=0$ or 1,

$$P(Sa|S)=\frac{P(S|Sa)P(Sa)}{P(S)}, \tag{3.1}$$

where $P(S)=P(S|S0)P(S0)+P(S|S1)P(S1)$. Since $P(S|Sa)=1$ for any a, we have,

$$P(Sa|S)=\frac{P(Sa)}{P(S0)+P(S1)}. \tag{3.2}$$

In terms of inductive inference or machine learning, the final probability $P(Sa|S)$ is the probability of the next symbol being a given the initial sequence S. The goal of inductive inference in general is to be able to infer the underlying machinery that generated S, and hence be able to predict (extrapolate) the next symbol. Obviously we now only need the prior probability $P(Sa)$ to evaluate $P(Sa|S)$. Let K denote the self-delimiting Kolmogorov complexity. Using the Solomonoff–Levin distribution, assign $2^{-K(Sa)}$ as the prior probability to $P(Sa)$. This corresponds to Occam's razor principle of choosing the simplest theory and has two very nice properties:

(1) $\sum_x 2^{-K(x)} \leqslant 1$ by Kraft's inequality. Hence this is a proper probability assignment.

(2) For any computable probability P, there is a constant c such that $2^{-K(x)} \geqslant c\,P(x)$, for all x. In fact, $\boldsymbol{m}(x)=2^{-K(x)\pm O(1)}$ [47, 173].

This approach is guaranteed to converge in the limit to the true solution and in fact it converges *faster than* any other method up to a constant multiplicative factor. It turns out that Gold's idea of identification by enumeration [53] can also be derived from this approach [96].

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 [24].

3.3. *Rissanen's Minimum Description Length Principle*

Scientists formulate their theories in two steps: first a scientist must, based on scientific observations or given data, formulate alternative hypotheses, and second he selects one definite hypothesis. This was done by many ad hoc principles, among the most dominant, Occam's razor principle, the maximum likelihood principle, various ways of using Bayesian formula with different prior distributions. However, no single principle is satisfactory in all situations. But in an ideal sense, the Solomonoff approach we have discussed presents a *perfect* way of solving all induction problems using Bayes' rule with the universal prior distribution. However, due to the noncomputability of the universal prior function, such a theory cannot be directly used in practice. Inspired by the Kolmogorov complexity research, in 1978 Rissanen proposed the *Minimum Description Length Principle* (*MDLP*) [132]. See also [177] for a related approach pioneered by C.S. Wallace and D.M. Boulton in 1968. Quinlan and Rivest used this principle to construct an algorithm for constructing decision trees and the result was quite satisfactory compared to existing algorithms [130]. Using MDLP, Gao and Li [51] have implemented a system (on IBM PC) which on-line recognizes/learns handwritten English and Chinese characters. The *Minimum Description Length Principle* can be intuitively stated as follows.

MINIMUM DESCRIPTION LENGTH PRINCIPLE. *The best theory to explain a set of data is the one which minimizes the sum of*

(1) *the length* (*encoded in binary bits*) *of the theory*;
(2) *the length* (*in binary bits*) *of data when encoded with the help of the theory.*

EXAMPLE (*Kemeny*). The importance of "simplicity" for inductive inference was already exploited in an elegant paper by Kemeny [70]. This paper clearly anticipates the ideas developed in this section, and we cite one example. We are given n points in the plane. Which polynomial fits these points best? One extreme is to put a straight line through the cluster such that the χ^2 measure is minimized. The other extreme is to simply fit an $n-1$ degree polynomial through the n points. Neither choice seems very satisfactory, and it is customary to think that the problem is not formulated precisely enough. But MDLP says that we look for the mth degree polynomial, $m \leqslant n-1$, such that the description of the m-vector of coefficients of the polynomial, together with the description of the points relative to the polynomial, is minimized.

It is remarkable that, using Kolmogorov complexity, we can formally derive a version of the MDL principle, and explain how and why it works [96]. From Bayes' rule,

$$P(H|D) = \frac{P(D|H)P(H)}{P(D)},$$

we need to choose the hypothesis H such that $P(H|D)$ is maximized, where D denotes the data. Now taking the negative logarithm on both sides of Bayes' formula, we get

$$-\log P(H|D) = -\log P(D|H) - \log P(H) + \log P(D).$$

Since D is fixed, maximizing the term $P(H|D)$ is equivalent to minimizing

$$-\log P(D|H) - \log P(H)$$

Now to get the minimum description length principle, we only need to explain the above two terms properly. The term $-\log P(H)$ is straightforward. Assuming the Solomonoff–Levin distribution, $P(H)=2^{-K(H)}$ where $K(H)$ is the self-delimiting Kolmogorov complexity of H. Then the term $-\log P(H)$ is precisely the *length* of a minimum-length *prefix-free encoding*, or shortest program, for the hypothesis H. Similarly for the term $-\log P(D|H)$.

Now just assume that P is computable. It can be shown that the universal probability distribution $\mathbf{m}(x)$ can approximate $P(x)$ in the sense that

(1) there is a constant c, such that $\mathbf{m}(x) \geqslant c\,P(x)$, and

(2) the probability that $\mathbf{m}(x) \leqslant k\,P(x)$ is at least $1-1/k$.

Since $\mathbf{m}(H)=2^{-K(H)\pm O(1)}$, the quantity $2^{-K(H)}$ is a reasonable estimate for $P(H)$. Similarly, $2^{-K(D|H)}$ is an estimate for $P(D|H)$. Hence we must minimize $K(D|H)+K(H)$, i.e., find an H such that the sum of the description lengths is minimized.

In the original Solomonoff approach, H in general is a Turing machine. In practice we must avoid such a too general approach in order to keep things computable. In different applications, the hypothesis H can mean many different things. For example, H may refer to decision trees, finite automata, Boolean formulae, or polynomials of certain degree. Thus, Rissanen suggested the following approach. First convert (or encode) H to an integer. Then we try to assign a prior probability to each integer. Assigning $2^{-K(n)}$ to integer n would be perfect but it is not computable. Jeffreys [67] suggested to assign probability $1/n$ to integer n. But this results in an improper distribution since $\sum_{n=1}^{\infty} 1/n$ diverges. Rissanen defined the following length function: let

$$l^*(n) = \log n + \log\log n + \log\log\log n + \cdots$$

all positive terms, and let $L(n)=l^*(n)+\log c$ where $c=2.865064\ldots$. It has been shown that $\sum_n 2^{-L(n)}=1$, see [133].

3.4. *Learnability in the Valiant learning model*

This section is rather closely related to the previous two sections. There we did not consider the number of steps involved in making an inference, or the number of examples needed to learn a concept. Solomonoff's principle shows that we learn something perfectly in the limit, but how fast this converges is not prescribed at the outset. For instance, the well-known principle of Gold [53] of inference by enumeration can be viewed as a particular case of Solomonoff's principle (cf. [96]) and we note here without further explanation, and as immediately clear to those familiar with it, that the Gold paradigm of inductive inference is aimed at *precise* inference. But what if we want to learn a concept using a number of examples that is bounded a priori? Obviously if we are to *precisely* infer a law of nature, 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 0.99 *probability*, the next apple after seeing three apples, we consider that the concept of apple is learned. In 1983, Valiant [157] introduced such a learning model. Valiant emphasizes polynomial-time learnability. For simplicity and convenience, we consider the problem of learning Boolean formulae of n variables.

According to Valiant, a concept F is, say, 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 least number of symbols needed to write such a 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^-). We assume nothing about the distributions D^+ and D^- except that $\sum_{f(v)=1} D^+(v)=1$ and $\sum_{f(v)=0} D^-(v)=1$. 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: $\sum_{g(v)=0} D^+(v)<\varepsilon$ and $\sum_{g(v)=1} D^-(v)<\varepsilon$.

Various classes of concepts are shown to be learnable in Valiant's sense [16, 58, 68, 99, 134, 157, 158]. Many Valiant learnable classes are NP-complete to learn in the Gold sense (see [69] for a survey). In [16], again by 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 (see also [96]). Many concepts turn out to be too hard (like NP-complete) to learn in Valiant's sense. A further refinement to learning "simple" concepts under all "simple" distributions, using the universal distribution $\boldsymbol{m}$, is developed in [176].

3.5. *Computable numbers are not random*

One can make precise the off-hand claim made in the Introduction that the set of computable numbers coincides with the set of reals which have Kolmogorov complexity $O(1)$. (We view a real number as an infinite sequence of digits.) Let $N=\{0,1,\ldots\}$ be the set of natural numbers, let $S=\{\varepsilon,0,1,01,10,11,\ldots\}$ 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 $s_{1:n}$ 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 $f\colon N\to S$ such that $x_{1:n}=f(n)$ for all n. Let K denote the plain Kolmogorov complexity as in Section 2. It can be shown [29] that x is recursive iff there exists a constant $c>0$ such that for all $n\in N$ we have $K(x_{1:n})\leqslant K(n)+c$.

3.6. *The number of wisdom Ω*

This Cabalistic exercise follows Chaitin [26, 27] and Bennett [11]. 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 [34]. 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 on the consecutive bits, can win infinite gain. Can we exhibit a specific such number? One can define an uncomputable number $k = 0.k_1 k_2 \ldots$ 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 (Section 2.5) k is *not* incompressible: each n-length prefix $k_{1:n}$ of k can be compressed to a string of length not more than $2 \log n$ (since $K(k_{1:n}|n) \leqslant \log n$), 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 [28] has found a number that is random in the strong sense needed.

Ω equals the probability that the reference self-delimiting universal Turing machine halts when its program is generated by fair coin tosses. That is, $\Omega = \sum_x \boldsymbol{m}(x)$, with $\boldsymbol{m}(x)$ the Solomonoff–Levin distribution. Then, Ω is a number between 0 and 1. It is greater than 0 since some programs do halt, and it is less than one since some programs do not halt (use the Kraft inequality). 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 $\Omega_{1:n}$ of Ω, then $\Omega_{1:n} < \Omega < \Omega_{1:n} + 2^{-n}$. However, dovetailing (execute phases $1, 2, \ldots$. where 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 $\Omega' > \Omega_{1:n}$. If p is not among the halted programs which contributed to Ω', then p will never halt, since otherwise its contribution would yield $\Omega \geqslant \Omega' + 2^{-n}$, which is a contradiction. (That is, $\Omega_{1:n}$ is a short program to obtain $k_{1:m}$ with $m \approx 2^n$.)

The argument suffices, via Barzdin's Theorem in Section 2.5, to establish that Ω is a random infinite sequence in the sense of Martin-Löf (Section 2.4).

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 counterexamples to Fermat's Last Theorem, Riemann's Hypothesis and most other conjectures in mathematics that can be refuted by single finite counterexamples. 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, $\Omega_{1:10,000}$ can be used to decide for every statement in the theory whether it is true, false or independent. Finally, knowledge of $\Omega_{1:n}$ suffices to determine whether $K(x) \leqslant n$ 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" [11].

EXAMPLE (*Chaitin*). Recall that the Barzdin'–Loveland Lemma states that for all r.e. sets each n-length initial segment of their characteristic sequence has Kolmogorov complexity $O(\log n)$. Kolmogorov has remarked that this implies that the solubility of the first n Diophantine equations in an effective enumeration can be decided using at most $O(\log n)$ bits extra information. Namely, given the number $m \leqslant n$ of soluble equations in the first n equations, we can find them all effectively in the obvious way. Chaitin observed that this is not the case if we replace the question of mere solubility by the question of whether there are finitely many or infinitely many nontrivially different solutions. Namely, no matter how many solutions we find for a given equation, by itself this can give no information on the question to be decided. It turns out that the set of indices of the Diophantine equations with infinitely many different solutions is not r.e. In particular, in the characteristic sequence each initial segment of length n has Kolmogorov complexity of about n. Chaitin says that this shows that randomness is inherent not only in natural phenomena (e.g., related to quantum mechanics), but also occurs in mathematics [32, 33]. More precisely, we have the following claim.

CLAIM. *There is an (exponential) Diophantine equation $A(n, x_1, x_2, \ldots, x_m) = 0$ which has infinitely many solutions $x_1, x_2, \ldots, x_m$ iff the n-th bit of Ω is* 1.

PROOF. By dovetailing the running of all programs of the reference self-delimiting machine U in the obvious way we find a computable sequence of rational numbers $r_1 \leqslant r_2 \leqslant \cdots$ such that $\Omega = \lim_{n \to \infty} r_n$. The set $R = \{(n, k)$: the nth bit of r_k is a $1\}$ is a recursively enumerable (even recursive) set. The main step is to use a theorem due to J.P. Jones and Y.V. Mateyasevich (*J. Symbol. Logic* **49** (1984), pp. 818–829) to the effect that "every recursively enumerable set R has a singlefold exponential Diophantine representation $A(p, y)$". That is, $A(p, y) = 0$ is an exponential Diophantine equation, and the singlefoldedness consists in the property that $p \in R$ iff there is a y such that $A(p, y) = 0$ is satisfied and, moreover, there is only a single such y. (Here both p and y can be multituples of integers; in our case p represents $\langle n, x_1 \rangle$, and y represents $\langle x_2, \ldots, x_m \rangle$. For technical reasons we consider as proper solutions only solutions x involving no negative integers.) Representing R this way, there is a Diophantine equation $A(n, k, x_2, \ldots, x_m) = 0$ which has exactly one solution $x_2, \ldots, x_m$ if the nth bit of the binary expansion of r_k is a one, and it has no solution $x_2, \ldots, x_m$ otherwise. Consequently, the number of different m-tuples $x_1, x_2, \ldots, x_m$ which are solutions to $A(n, x_1, x_2, \ldots, x_m) = 0$ is infinite if the nth bit of the binary expansion of Ω is a 1, and this number is finite otherwise. $\square$

4. Example of an 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 that Chaitin gives of Euclid's theorem that the number of primes is infinite [31] yields a very simple proof of a weak prime number theorem. Let $\pi(n)$ denote the number of prime numbers less than n. (Recall that $\pi(n)$ is asymptotically $n/\log n$). We prove that $\pi(n)$ is $\Omega(\log n(\log\log n)^{-1})$. Let n be a random number with $K(n) \geqslant \log n - O(1)$. Consider a prime factorization

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

with $p_1, p_2, \ldots$ 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 $p_i \geqslant p_1 = 2$, it holds that $e_i \leqslant \log n$ and, bounding $K(e_i)$ by the length of self-delimiting descriptions of e_i,

$$K(e_i) \leqslant \log \log n + 2 \log \log \log n,$$

for all $i \leqslant m$. Therefore,

$$K(n) \leqslant \pi(n)(\log \log n + 2 \log \log \log n).$$

Substituting the lower bound on $K(n)$, we obtain the claimed lower bound on $\pi(n)$ for the special sequence of random n.

Recently, P. Berman [Personal communication] obtained the stronger result that the number of primes below n is $\Omega(n/\log^2 n)$, by an elementary Kolmogorov complexity argument. It is interesting because it shows a relation between primality and prefix codes. Recall that we identify the positive integer n with the nth binary string. Assume that we have a function $c: N \to N$ with the following property: for every two integers m, n, $c(m)$ is not a prefix of $c(n)$. Then c is called a *prefix code*. Consider only prefix codes c such that $c(n) = o(n^2)$. (For instance, choose $c(n) = \overline{|n|}n$, the self-delimiting description of n with binary length $|c(n)| = \log n + 2 \log \log n$ bits.)

Lemma (Berman). *For an infinite subsequence of positive integers n, $p_n = O(c(n))$, where p_n is the n-th prime.*

Choosing $c(n)$ as above we have $c(n) \leqslant n \log^2 n$. Therefore, by the lemma, p_n is $O(n \log^2 n)$. Straightforward manipulation of the order-of-magnitude symbols then shows that $\pi(n)$ is $\Omega(n/\log^2 n)$. This can be strengthened by choosing more efficient codes, but not all the way to obtain $\pi(n) = \Omega(n/\log n)$.

5. Applications of incompressibility: proving lower bounds

It was observed in [126] that the 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. A traditional lower bound proof by counting 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. 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. 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 subsection, 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 ten 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. The questions addressed in the next few subsections often deal with simulating one machine model by another, e.g., as treated in P. van Emde Boas' Chapter 1 "Machine Models and Simulations" in this Handbook.

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.

5.1.1. *Example 1: one-tape Turing machines*

Consider a most basic Turing machine model with only one tape, with a two-way read/write head, which serves as both input and worktape. The input is initially put onto the first n cells of the only tape. We refer a reader not familiar with Turing machines to [62] for a detailed definition. The following theorem was first proved by Hennie and a proof by counting, for comparison, can be found in [62, page 318]. In [124] the following elegant proof is presented. 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 $L=\{ww^{\mathrm{R}}: w\in\{0,1\}^*\}$ (the palindromes). Similarly for $L'=\{w2w^{\mathrm{R}}: w\in\{0,1\}^*\}$.*

PROOF (*cf.* [124]). Assume on the contrary that M accepts L in $o(n^2)$ time. Let $|M|$ denote the length of the description of M. 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 states the finite control is in when the tape head crosses the intersquare boundary between this square and its left neighbor. If cs is a crossing sequence, then $|cs|$ denotes the length of its description. Consider an input of $w0^nw^{\mathrm{R}}$ of length $3n$. Divide the tape segment containing the input into three equal-length segments of size n. If each crossing sequence associated with a square in the middle segment is longer than $n/(10|M|)$ then M spent $\Omega(n^2)$ time on this input. Otherwise there is a crossing sequence of length less than $n/(10|M|)$. Assume that this occurs at c_0. Now this crossing sequence requires at most $n/10$ bits to encode. Using this crossing sequence, we reconstruct w as follows. For every string $x0^nx^R$ of length $3n$, put it 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 $w=x$. Otherwise, if $w \neq x$ and the crossing sequences are consistent in both computations, then M accepts a wrong input $x0^nw^R$. However, this implies

$$K(w)<|cs|+O(\log n)<n,$$

contradicting $K(w)\geqslant n$. □

5.1.2. Example 2: parallel addition

Consider the following widely used and most general parallel computation model, the priority PRAM. A priority PRAM consists of processors $P(i)$, $i=1,2,\ldots,n^{O(1)}$, and an infinite number of shared memory cells $C(i)$, $i=1,2,\ldots$ 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 [59] using a Ramsey theorem, and in [66, 122]. 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 [93]. Beame [10] obtained a different proof, independently.

PROOF (*cf.* [93]). 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 $K^A(X)\geqslant|X|$. Divide X equally into n parts $x_1,x_2,\ldots,x_n$. Then consider the (*fixed*) computation of M on input $(x_1,\ldots,x_n)$. 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 $t'\geqslant t$ which is read at some step $t''\geqslant t'$ 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 that 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 $O(2^T \log n^{O(1)}) = o(n \log n)$ 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 of 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 $K^A(X) \leqslant J < n^3$. This contradicts the randomness of X. □

5.1.3. Example 3: Boolean matrix rank (Seiferas–Yesha)

For all n, there is an n by n matrix over GF(2) (a matrix with zero-one entries with the usual Boolean multiplication and addition) such that every submatrix of s rows and $n-r$ columns ($r, s \leqslant \frac{1}{4}n$) has at least $\frac{1}{2}s$ linear independent rows.

REMARK. Combined with the results in [20, 172] this example implies that $TS = \Omega(n^3)$ is an optimal lower bound for Boolean matrix multiplication on any general random-access machines, where T stands for time and S stands for space.

PROOF. Fix a random sequence x of elements in GF(2) (zeros and ones) of length n^2, so $K(x) \geqslant |x|$. Arrange the bits of x into a matrix M, one bit per entry in, say, the rowmajor order. We claim that this matrix M satisfies the requirement. To prove this, suppose this is not true. Then consider a submatrix of M of s rows and $n-r$ columns, $r, s \leqslant \frac{1}{4}n$. Suppose that there are at most $\frac{1}{2}s - 1$ linearly independent rows. Then $1 + \frac{1}{2}s$ rows can be expressed by the linear combination of the other $\frac{1}{2}s - 1$ rows. Thus we can describe this submatrix using

- The $\frac{1}{2}s - 1$ linear independent rows, in $(\frac{1}{2}s - 1)(n-r)$ bits;
- for each of the other $\frac{1}{2}s + 1$ rows, use $(\frac{1}{2}s - 1)$ bits.

Then to specify x, we only need to specify, in addition to the above, (i) M without the bits of the submatrix, and (ii) the indices of the columns and rows of this submatrix. When we list the indices of the rows of this submatrix, we list the $\frac{1}{2}s - 1$ linearly independent rows first. Hence we only use

$$n^2 - (n-r)s + (n-r)\log n + s \log n + (\tfrac{1}{2}s - 1)(n-r) + (\tfrac{1}{2}s - 1)(\tfrac{1}{2}s + 1) < n^2$$

bits, for large n's. This contradicts the fact $K(x) \geqslant |x|$. □

REMARK. 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 in Section 5.7.

5.2. Lower bounds: more tapes versus fewer tapes

Although Barzdin [9] and Paul [124] are the pioneers of using Kolmogorov complexity to prove lower bounds, the most influential paper is probably the one by Paul, Seiferas and Simon [126], which was presented at the 1980 STOC. This was partly because [124] was not widely circulated and, apparently, the paper by Barzdin [9] did not even reach this community. The major goal of [126] was "to promote the approach" of applying Kolmogorov complexity to obtain lower bounds. In [126], 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 $k \geqslant 1$ far more powerful than the model of Example 1, where the single tape is both input tape and worktape. For instance, a one-(work)tape Turing machine can recognize the marked palindromes of Example 1 in real time $T(n)=n$ in contrast with $T(n)=\Omega(n^2)$ required in Example 1.

In 1982 Paul [127], using Kolmogorov complexity, extended the results in [126] 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. Duriš, Galil, Paul and Reischuk [42] then improved the lower bound for the one- versus two-tape case to $\Omega(n \log n)$.

To simulate k tapes with 1 tape, the known (and trivial) upper bound on the simulation time was $O(n^2)$. The above 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 [163] an $\Omega(n^{1.5})$ 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 papers [164, 165] and the optimal results cited hereafter. Around 1983/1984, independently and in chronological order[2], Wolfgang Maass at UC Berkeley, the first author of the present chapter 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 one tape versus

[2] *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. Computer Science, 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' and Li's result were for off-line computation with one-way input, while Vitányi's result was for on-line computation. Li and Vitányi combined these and other results in [94], while Maass published in [105].

k (w.l.o.g. two) tapes. These lower bounds, and the following ones, were proven with as simulator an *off-line* machine with *one-way* input. All three relied on Kolmogorov complexity, and actually proved more in various ways[2]. Thus, Maass also obtained a nearly optimal result for nondeterministic simulation: [105] 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 [94] to

$$\Omega(n^2/\log n \log\log n)$$

time using Maass' language, and by Galil, Kannan and Szemeredi [49] to $\Omega(n^2/\log^{(k)}n)$ (for any k, with $\log^{(k)}$ 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 [94] presented the following lower bounds, all by Kolmogorov complexity. To simulate two pushdown stores, or only one queue, by one deterministic tape requires $\Omega(n^2)$ time. Both bounds are tight. (Note that the two-pushdown-stores result implies the two-tape result. However, the one-queue result is incomparable with either of them.) Further, one-tape nondeterministic simulation of two pushdown stores requires $\Omega(n^{1.5}/\sqrt{\log n})$ time. This is almost tight because of [89]. Finally, one-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 is $O(n^{1.5}\sqrt{\log n})$ in [89]. In a successor paper, together with Longpré, we have extended the above work with a comprehensive study stressing queues in comparison to stacks and tapes [91]. 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 one pushdown store (and hence one tape) by one queue requires $\Omega(n^{4/3}/\log n)$, in both the deterministic and nondeterministic cases. Simulation of one queue by one tape was resolved above, and simulation of one queue by one pushdown store is trivially impossible. Nondeterministic simulation of two queues (or two tapes) by one queue requires $\Omega(n^2/(\log^2 n \log\log n))$ time, and deterministic simulation of two queues (or two tapes) by one queue requires quadratic time. All these results would be formidable without Kolmogorov complexity.

A next step is to attack the similar problem with a *two-way input* tape. Maass and Schnitger [106] proved that when the input tape is two-way, two worktapes are better than one 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 worktape(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. In [106] it is shown that transposing a matrix (with element size $O(\log n)$) requires $\Omega(n^{3/2}(\log n)^{-1/2})$ time on a one-tape off-line Turing machine with an extra two-way read-only input tape. The first version of this paper (authored by Maass alone) 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 [106].) This gives the desired separation of two tapes versus one, because with two worktapes, 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 [107] in 1987 finally resolved the question of whether two tapes are better than one with two-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 non-zeros): $\{A^{**}B: A = B^t$ 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 [107] and the $O(n^2)$ upper bound. In [106] it was observed that if the Turing machine has a one-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 worktape and no output tape, once some bits have been written they can later be moved only by time-wasting sweeps of the worktape head. In contrast, with an output tape, as long as the output data are computed in the correct order, they can be output and do not have to be moved again. Using Kolmogorov complexity, in [41] Dietzfelbinger shows that transposition of Boolean matrices by Turing machines with two-way input tape, one worktape, and a one-way output tape requires $\Omega(n^{5/4})$ time, thus matching the upper bound for matrix transposition.

It turns out that the similar lower bound results for higher-dimensional tapes are also tractable, and sometimes easier to obtain. The original paper [126] contains such lower bounds. M. Loui proved the following results by Kolmogorov complexity. A *tree worktape* is a complete infinite rooted binary tree as storage medium (instead of a two-way infinite linear tape). A worktape head starts at the origin (the root) and in each step can move to the direct ancestor of the currently scanned node (if it is not the root) or to either one of the direct descendants. A *multihead tree machine* is a Turing machine with a tree worktape with $k \geqslant 1$ tree worktape heads. We assume that the finite control knows whether two worktape heads are on the same node or not. A *d-dimensional worktape* consists of nodes corresponding to d-tuples of integers, and in each step a worktape head can move from its current node to a node with each coordinate ± 1 of the current coordinates. Each worktape head starts at the origin which is the d-tuple with all zeros. A *multihead d-dimensional machine* is a Turing machine with a d-dimensional worktape with $k \geqslant 1$ worktape heads. M. Loui [102] has shown that a multihead d-dimensional machine simulating a multihead tree machine on-line (both machines have a one-way input tape and one-way output tape) requires time $\Omega(n^{1+1/d}/\log n)$ in the worst case, and he proved the same lower bound for the case where a multihead d-dimensional machine is made more powerful by allowing the worktape heads also to move from their current node to the current node of any other worktape head in a single step. The lower bound is optimal.

5.3. *Lower bounds: more heads versus fewer heads*

Again applying Kolmogorov complexity, Paul [125] showed that two-dimensional two-tape (with one head on each tape) Turing machines cannot simulate on-line two-dimensional Turing machines with two heads on one tape in real time. He was not able to resolve this problem for one-dimensional tapes, and, despite quite some effort, the following problem is open and believed to be difficult: Are two heads on one (one-dimensional) tape better than two (one-dimensional) tapes, each with one head? The following result, proved using Kolmogorov complexity, is intended to be helpful in separating these classes. A Turing machine with two one-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 [162]. (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, Seiferas [142] 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 1960s and 1970s. 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 *one-way* heads on the input tape. Two rather basic questions were left open from the automata and formal language theory of the 1960s:

(1) Rosenberg Conjecture (1965): $(k+1)$-head finite automata are better than k-head finite automata [136, 137].

(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 [55].

In 1965, Rosenberg [137] claimed a solution to problem (1), but Floyd [44] pointed out that Rosenberg's informal proof was incomplete. In 1971 Sudborough [152, 153], and later Ibarra and Kim [65] obtained a partial solution to problem (1) for the case of two heads versus three heads, with difficult proofs. In 1976 Yao and Rivest [171] finally presented a full solution to problem (1). A different proof was also obtained by Nelson [119]. Recently it was noted by several people, including Seiferas and the present authors, that the Yao–Rivest proof can be done very naturally and easily by Kolmogorov complexity: Let

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

as defined by Rosenberg and Yao–Rivest. Let $b=\binom{k}{2}+1$. 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 $w_1w_2\dots w_b$. We say that the two w_i's in L_b are matched if there is a time such that two heads of M are in the two 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 from the positions of heads and states for M when a head comes in/out w_i; $K(w_i|W-w_i)=O(k\log n)<\frac{1}{2}|w_i|$, a contradiction.

The Harrison–Ibarra Conjecture, however, was open until the time of applied Kolmogorov complexity. Several authors tried to generalize the Yao–Rivest method [116, 117] or the Ibarra–Kim method [35] to the k-PDA case, but only partial results were obtained. For the complete odyssey of these efforts see the survey in [36]. With the help of Kolmogorov complexity, [36] presented a complete solution to the Harrison–Ibarra Conjecture for the general case. The proof was constructive, 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 [48]. They proved that a six-head *two-way* DFA can do string (pattern) 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 Seiferas suggested the use of Kolmogorov complexity, which shortened the proof to less than a page [92]! By similar methods a proof that 3-DFA cannot do string matching was also obtained [90].

5.4. Lower bounds: parallel computation and branching programs

In Example 2 we have seen 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.

Fast addition or multiplication of n numbers in parallel is obviously important. In 1985 Meyer auf der Heide and Wigderson [59] proved, using Ramsey theorems, that on a 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 [66] and Parberry [122]. 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 [59]. In [93] Kolmogorov complexity is applied to obtain parallel lower bounds (and trade-offs) 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 [59, 66, 122]. 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 $\{\text{AND}, \text{OR}, \text{NOT}\}$ 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. Without loss of generality, let the first level of C be AND gates and the second level be an OR gate.

Consider an $x=x_1 \ldots x_n$ such that $K(x|C) \geqslant |x| = n$ 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 $\log_2(o(2^n))$. 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 $\overline{x_i}$ for some i. Hence changing only the value of x_i in x does not change the output (value 1) of G and C, a contradiction. (Note that 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 [124].) In 1979 Borodin, Fischer, Kirkpatrick, Lynch and Tompa proved a time–space trade-off for comparison-based sorting algorithms [19]. This was improved and generalized to a very wide class of sequential sorting algorithms by Borodin and Cook [20] defined as "branching programs". The proof involved difficult counting. In [131] Reisch and Schnitger used Kolmogorov complexity, in one of their three applications, to simplify the well-known $\Omega(n^2/\log n)$ bound of Borodin and Cook [20] for the time–space trade-off in sorting with branching programs. They also improved the lower bound in [20] to $\Omega(n^2 \log\log n/\log n)$.

5.5. *Lower bounds: time–program size trade-off for searching a table*

"Is x in the table?" Let the table contain n keys. You can sort the table and do binary search on the table; then your program can be as short as $\log n$ bits, but you use about $\log n$ time (probes). Or you can do hashing; you can use a perfect hashing function $h(x)=[A/(Bx+C)]$ [23, 112]; then your program can be as long as $\Omega(n)$ bits since A, B, C need to have $\Omega(n)$ bits to make $h(x)$ perfect, but the search time is $O(1)$ probes. What is the size of the program? It is nothing but the Kolmogorov complexity.

Searching a table is one of the most fundamental issues in computer science. In a beautiful paper [108] Mairson literally studied *the program complexity* of table-searching procedures in terms of the number of bits that is required to write down such programs. In particular, he proved that a perfect hashing function of n keys needs $\Theta(n)$ bits to implement. He also provided the trade-offs between the time needed to search a table and the size of the searching program.

5.6. *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 of arguments similar to that of Turing machines [98]. 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 [98], which is a generalization of Thompson's model [154]. 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 $f(x_1, x_2, \ldots, x_n)$ which is to be computed;

(b) a synchronous circuit C, computing 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); and

(c) 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 [154] and later generalized in [98].

We prove $AT^2 = \Omega(n^2)$ lower bounds for many problems roughly as follows. Draw a line to divide the layout into two parts, with about half the 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 two parts. Then for each specific problem one only needs to show that $I = \Omega(n)$ for any division. Lipton and Sedgewick defined a *crossing sequence* to be, roughly, the sequence of T tuples $(v_1, \ldots, v_\omega)$ where the ith tuple contains the values appearing 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 nonintuitive 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 [98] 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 [98]. A sample question was formulated in [98].

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 two 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 $AT^2 = \Omega(n^2)$. $\square$

All other problems, selection/equality testing, DCFL, factor verification, listed in [98] can be done similarly, even under the nondeterministic, or randomized, circuits as defined in [98].

Some general considerations on VLSI lower bounds using Kolmogorov complexity were given by R. Cuykendall [39]. L.A. Levin and G. Itkis have investigated the VLSI computation model under different information transmission assumptions [87, 175]. In their model, if the speed of information transmission is superlinear, namely $\max(K(d) - \log f(d)) < \infty$ with $f(d)$ 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 considerable savings in required area). Note that $f(d) = \Omega(d \log^2 d)$ satisfies the requirements for f, but not $f(d) = O(d)$.

5.7. Lower bounds: randomized algorithms

We have seen that Kolmogorov complexity can be naturally applied to nondeterministic Turing machines. Hence, it is likely that it is useful for analyzing randomized algorithms. Indeed this is the case. In their paper about three applications of Kolmogorov complexity [131] Reisch and Schnitger analyzed, using Kolmogorov complexity, the probabilistic routing algorithm in the n-dimensional binary cubes of Valiant and Brebner [156].

In 1983 Paturi and Simon generalized the deterministic lower bounds previously proved in [1, 125–127, 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 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. A version of the Symmetry of Information Lemma stated in Sections 2.6 and 2.7 is proved in [123]. They show that for a sequence of random coin tossing, the probability that this sequence β of random coin tossing bits 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 2). Note that many such α's 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 in 1974 by Levin who called the general principle involved "Law of Information Conservation" [86]; see for later developments also [88].

5.8. Lower bounds: formal language theory

The classic introduction to formal language theory is [62]. 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" lemmas, i.e., the *uvw*-lemma (*uvwxy*-lemma respectively). However, these lemmas 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 material from our paper [95]. Without loss of generality, 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 an alphabet V consists of the union of some equivalence classes of a right-invariant equivalence relation on V^* $(=\bigcup_{i\geqslant 0} V^i)$ of finite index. Let us give an example of how to use Kolmogorov complexity to prove nonregularity. We prove that $\{0^k1^k : k \geqslant 1\}$

is not regular. To derive a contradiction, suppose it is regular. Fix k with $K(k) \geqslant \log k$, 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 ones, it reaches its first accepting state precisely after k ones. 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 y is the n-th string in the lexicographical order in $L_x = \{y: xy \in L\}$ (or in the complement of L_x) then $K(y) \leqslant K(n) + c$.*

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 and gives a Kolmogorov complexity characterization of regular languages [95]. 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, as shown in the following example.

EXAMPLE. We prove that $\{1^p : p \text{ is prime}\}$ is not regular. Consider the string xy consisting of p ones, with p the $(k+1)$st prime. In the lemma set 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 unboundedly, 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 unboundedly is as follows. Let P be the product of the first j primes. Clearly, no $P+i$, $1 < i \leqslant j$, is prime.)

EXAMPLE. (*cf.* [62, *Exercise* 3.1(h*)]). Prove that $L = \{xx^Rw : x, w \in \{0,1\}^* - \{\varepsilon\}\}$ is not regular. Set $x = (01)^n$, where $K(n) \geqslant \log n$. Then the lexicographically first word in L_x is $y = (10)^n 0$. Hence, $K(y) = \Omega(\log n)$, contradicting the KC-regularity Lemma.

EXAMPLE (*cf.* [62, *Exercise* 3.6*]). Prove that $L = \{0^i 1^j : GCD(i,j) = 1\}$ is not regular. For each prime p, the string $0^p 1^p$ is the second word with prefix 0^p. Hence by the KC-regularity Lemma there is a constant c such that for all p we have $K(p) < c$, which is a contradiction.

Similar general lemmas can also be proved to separate DCFLs from CFLs. Previous proofs that a CFL is not a DCFL often use ad hoc methods. We refer the interested readers to [95].

Concerning formal language theory we must mention a beautiful proof due to Seiferas. It is known that linear context-free languages can be recognized on-line by a one-worktape Turing machine in $O(n^2)$ time. This result is due to Kasami. Gallaire, using a very complicated counting argument and *de Bruijn sequences* [50], proved that a multitape Turing machine requires $\Omega(n^2/\log n)$ time to recognize on-line linear context-free languages. In [143] Seiferas presented an elegant and short proof of the same bound using Kolomogorov complexity, significantly simplifying Gallaire's proof.

5.9. Lower bounds: which method to use?

Instead of attempting to answer this difficult question, we present a problem with three proofs: one by counting, one by probabilistic argument and one by Kolmogorov complexity. The question and first two proofs are taken from a beautiful book by Erdös and Spencer [43].

A *tournament* T is a complete directed graph, i.e., for each pair of vertices u and v in T, exactly one of the edges (u, v), (v, u) is in the graph. Given a tournament T of n nodes $\{1, \ldots, n\}$, fix any standard effective coding, denoted by $c(T)$, using $\frac{1}{2}n(n-1)$ binary bits, one bit for each edge. The bit of edge (u, v) is set to 1 iff $u < v$. The next theorem and the first two proofs are from the first example in [43].

THEOREM. *If $v(n)$ is the largest integer such that every tournament on $\{1, \ldots, n\}$ contains a transitive subtournament on $v(n)$ players, then $v(n) \leqslant 1 + [2 \log_2 n]$.*

REMARK. This theorem was proved first by Erdös and Moser in 1964. Stearns showed by induction that $v(n) \geqslant 1 + [\log_2 n]$.

PROOF (*by counting*). Let $v = 2 + [2 \log_2 n]$. Let $\Gamma = \Gamma_n$ be the class of all tournaments on $\{1, \ldots, n\}$, and Γ' the class of tournaments on $\{1, \ldots, n\}$ that do contain a transitive subtournament on v players. Then

$$\Gamma' = \bigcup_A \bigcup_\sigma \Gamma_{A,\sigma} \tag{5.1}$$

where $A \subseteq \{1, \ldots, n\}$, $|A| = v$, σ is a permutation on A, and $\Gamma_{A,\sigma}$ is the set of T such that $T|A$ is generated by σ. If $T \in \Gamma_{A,\sigma}$, the $\binom{v}{2}$ games of $T|A$ are determined. Thus

$$|\Gamma_{A,\sigma}| = 2^{\binom{n}{2} - \binom{v}{2}} \tag{5.2}$$

and by elementary estimates

$$|\Gamma'| \leqslant \sum_{A,\sigma} 2^{\binom{n}{2} - \binom{v}{2}} = \binom{n}{v} v! 2^{\binom{n}{2} - \binom{v}{2}} < 2^{\binom{n}{2}} = |\Gamma|. \tag{5.3}$$

Thus $\Gamma - \Gamma' \neq \emptyset$. That is, there exists $T \in \Gamma - \Gamma'$ not containing a transitive subtournament on v players. □

PROOF (*by probabilistic argument*). Let $v = 2 + [2 \log_2 n]$. Let $\Gamma = \Gamma_n$ be the class of all tournaments on $\{1, \ldots, n\}$. Let also $A \subseteq \{1, \ldots, n\}$, $|A| = v$, and let σ be a permutation on A. Let $\boldsymbol{T} = T_n$ be a random variable. Its values are the members of Γ where, for each $T \in \Gamma$, $\Pr(\boldsymbol{T} = T) = 2^{-\binom{n}{2}}$. That is, all members of Γ are equally probable values of $\boldsymbol{T}$.

Then

$$\begin{aligned}&\Pr(\boldsymbol{T}\text{ contains a transitive subtournament on } v \text{ players})\\ &\quad\leqslant \sum_{A}\sum_{\sigma}\Pr(\boldsymbol{T}\,|\,A \text{ generated by } \sigma)\\ &\quad=\binom{n}{v}\, v!\,2^{-\binom{v}{2}}<1.\end{aligned}$$

Thus some value T of $\boldsymbol{T}$ does not contain a transitive subtournament on v players. □

Proof (*by Kolmogorov complexity*). Fix $T\in\Gamma_n$ such that

$$K(c(T)\,|\,n, v(n))\geqslant |c(T)| = \tfrac{1}{2}n(n-1).$$

Suppose $v(n)=2+[2\log_2 n]$ and let S be the transitive tournament of $v(n)$ nodes. We effectively recode $c(T)$ as follows in less than $|c(T)|$ bits, and hence we obtain a contradiction, by

- listing in order of dominance the index of each node in S in front of $c(T)$, using $2(\lceil\log_2 n\rceil)^2+2\lceil\log_2 n\rceil+|c(T)|$ bits;
- deleting all bits from $c(T)$ for edges in between nodes in S to save $2(\lceil\log_2 n\rceil)^2+3\lceil\log_2 n\rceil+1$ bits. □

5.10. *Lower bounds: open questions*

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

(1) Can k-DFA do string matching [48]?

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

(3) Prove a tight, or $\Omega(n^{1+\epsilon})$, lower bound for simulating two tapes by one for off-line Turing machines with an extra two-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 devices used to compress strings. Says Kolmogorov [76] in 1965:

> "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 [9] cited earlier. Time-limited Kolmogorov complexity was applied by Levin [84] in relation with his independent work on NP-completeness, and further studied in [88]. Adleman investigated such notions [2], in relation to factoring large numbers. Resource-bounded Kolmogorov complexity was extensively investigated by Daley [40], Hartmanis [56], and Ko [73]. Sipser [146] 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: $\mathrm{BPP} \leqslant \Sigma_4 \cap \Pi_4$. (Gács improved this to $\mathrm{BPP} \leqslant \Sigma_2 \cap \Pi_2$.) We treat the more influential approaches of Adleman, Bennett, Hartmanis and Sipser 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 [88] and Bennett [12].

6.1. *Potential*

In an elegant paper [2], Adleman formulates the notion of *potential* as the amount of time that needs to be pumped into a number by the computation that finds it. That is, 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 a lot of time. Is there a notion of storing potential in numbers with the result that high-potent primes have relatively low-potent 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 (the class of problems that can be solved by deterministic algorithms in time polynomial in the input length) then this is the reason why. Formally, we use the following definition.

DEFINITION. For all integers $k \geqslant 0$, 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 $\leqslant k \log|x|$ 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 x can mean the positive integer x or the xth binary string.)

EXAMPLE. For almost all $n \in N$, 1^n is 2-potent. Namely, $|1^n| = n$ and $|n| \sim \log n$. 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 all k, the function*

$$f_k(x, 1^n) = \{y : |y| \leqslant n \text{ and } y \text{ is } k\text{-potent w.r.t. } x\}$$

is computable in polynomial time.

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

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 products.*

THEOREM (Adleman). *With respect to the* $\mathrm{P} = \mathrm{NP}$ *question:* $SAT \in \mathrm{NP} - \mathrm{P}$ *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*

Bennett has formulated an intriguing notion of logical depth [12–14]. 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 Gács' reformulation, using the Solomonoff–Levin approach to a priori probability $\boldsymbol{m}$, cf. Section 2.9),

$$depth_\varepsilon(x) = \min\{t : \boldsymbol{m}_t(x)/\boldsymbol{m}(x) \geqslant \varepsilon\}.$$

Here, $\boldsymbol{m}_t$ is the t-bounded analogue of $\boldsymbol{m}$. 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, see [12] or the example below.) According to Bennett, quoted in [30]: "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 $\leqslant d$ 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 be quickly 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*). This example concerns the distinction between depth and information: consider the numbers k and Ω (see Section 3.6). 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 $n+\mathrm{O}(\log n)$ 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. That is, Barzdin' [9] showed that the initial segments of k are compressible to the logarithm of their length if unlimited time is allowed for decoding, but they can only be compressed by a constant factor if any recursive bound is imposed on the decoding time. The precise statement of this is given at the end of Section 2.5.

6.3. *Generalized Kolmogorov complexity*

Below we partly follow [56]. Assume that we have fixed a universal Turing machine U with an input tape, worktapes and an output tape. "A string x is computed from a string z" (z is a *description* of x) means that U starting 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 Kolmogorov complexity as described in Section 27.

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 ${}^{\mathrm{UP}}_{U}(K, T, S)$ if x can be computed from a string z, $|z| \leqslant K \leqslant |x|$, in $\leqslant T$ steps by U using $\leqslant S$ space on its worktape. A string x of length n is in complexity class $K^{\mathrm{UP}}_{U}[k(n), t(n), s(n)]$ if $K \leqslant k(n)$, $T \leqslant t(n)$ and $S \leqslant s(n)$. Thus, we consider a computation that *inflates* z to x. A string x has *resource-bounded* Kolmogorov complexity ${}^{\mathrm{DOWN}}_{U}(K, T, S)$ if some description z of x can be computed from x, $|z| \leqslant K \leqslant |x|$, in $\leqslant T$ steps by U using $\leqslant S$ space on its worktape. Here we consider a computation that *deflates* x to z. A string x of length n is in complexity class $K^{\mathrm{DOWN}}_{U}[k(n), t(n), s(n)]$ if $K \leqslant k(n)$, $T \leqslant t(n)$ and $S \leqslant s(n)$. Clearly,

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

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

It follows immediately by the Hennie–Stearns simulation of many worktapes by two worktapes (cf. [62]) that there is a U with two worktapes such that, for any multitape universal Turing machine V, there is a constant c such that

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

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,

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

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

The obvious relation between inflation and deflation is

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

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

(there are at most $2^{k(n)}$ (resp. 2^n) possibilities to try). Our $K^{\mathrm{UP}}[f(n), g(n), h(n)]$ will also be written as $K[f(n), g(n), h(n)]$ to be consistent with literature as in [56].

In his Ph.D. Thesis [101], Longpré analyzed 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 pseudorandom number generator that passes Yao's test [170].

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

Example (*Computation time*). Related to the notions potential and logical depth is Levin's concept of *time of computation complexity Kt* [88]. In this framework we formulate it as follows: $x \in \{0,1\}^*$ has Kt-complexity $Kt(x) = m$ if $x \in K[m - \log t, t, \infty]$, m minimal.

Example (*Hartmanis*). The sparse set

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

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

In [56] 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 [15]. In [57] a characterization of the P = NP question is given in terms of time-bounded Kolmogorov complexity and relativization. Earlier, Adleman in [2] established a connection, namely NP $\neq$ P exactly when NP machines can "manufacture" randomness. Following this approach, Hemachandra [60] obtains unrelativized connections in the spirit of [57].

Example (*Hartmanis*). Hartmanis noticed the following interesting fact: a polynomial machine cannot from simple input compute 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 $C \subseteq \{1^{2^n}: n \geqslant 1\}$ and $C \in \mathrm{DTIME}(n^{\log n}) - P$. For every n such that $1^{2^n} \in C$, put the first string of length 2^n from

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

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.

Example (*Longpré, Natarajan* [101, 118]). It was noticed that Kolmogorov complexity can be used to obtain space complexity hierarchies in Turing machines. Also it can be used to prove certain immunity properties. For example, one can prove that if $\lim_{n\to\infty} S(n)/S'(n) = 0$, then, for any universal machine U, if $S'(n) \geqslant n$ is a nondecreasing function and if $f(n)$ is a function not bounded by any constant and computable in space $S(n)$ by U, then we have that the complement of $K_U[f(n), \infty, S'(n)]$ is DSPACE($S(n)$)-immune for large n.

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 a previous section. Other work in this area includes [5, 8, 128]. 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 $E = \mathrm{DTIME}(2^{cn})$. Book, Orponen, Russo, and Watanabe [18] constructed an exponentially low set A which is not in P. We give this elegant construction in detail. Let $K = K[\frac{1}{2}n, 2^{3n}, \infty]$ and $\bar{K}$ its complement. Let $A = \{x: x$ is the lexicographically least element of $\bar{K}$ of length $2^{2^{\cdot^{\cdot^{\cdot^2}}}}$ (stack of m 2s), 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 we do the following: If a query to A is of correct length (stack of 2s) 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 2s) is not in A and

(2) a string of length greater than cn is not even in $\bar{K}$.

(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, $\frac{1}{4}cn$ bits assuming c is chosen properly; therefore the query string is in K.

In [168], 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 [6] used Kolmogorov complexity to characterize the class of sets which are polynomial many-one equivalent to tally sets, in order to study the question of whether $E^{\mathrm{P}}_{\mathrm{m}}(Tally)=E^{\mathrm{P}}_{\mathrm{btt}}(Tally)$ is true, where $E^{\mathrm{P}}_{-}(Tally)=\{L\colon \text{for some tally set } T,\ L\equiv^{\mathrm{P}}_{-} T\}$. In [63, 64], Huynh started a series of studies on the concept of resource-bounded Kolmogorov complexity of languages. He defined the (time/space-bounded) Kolmogorov complexity of a language to be the (time/space-bounded) Kolmogorov complexity of

$$Seq(L^{<n})=C_L(w_1)C_L(w_2)\ldots C_L(w_{2^n-1}),$$

where w_i is lexicographically the ith word and $C_L(w_i)=1$ iff $w_i\in L$. In particular, he shows that there is a language $L\in \mathrm{DTIME}(2^{2^{O(n)}})$ (any hard set for this class) such that the 2^{Poly}-time-bounded Kolmogorov complexity of L is exponential almost everywhere. That is, 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. Compare this with Barzdin's result, cited at the end of Section 2.5.

Allender and Rubinstein studied the relation between small resource-bounded Kolmogorov complexity and P-printability. Sets like $K[k\log n, n^k, \infty]$ for some constant k are said to have small time-bounded Kolmogorov complexity. A set S is said to be polynomial-time printable (P-printable) if there is a k such that all the elements of S up to size n can be printed by a deterministic machine in time n^k+k. Clearly, every P-printable set is a sparse set in P. Define the *ranking function* for a language L, $r_L:\Sigma^*\to N$, given by $r_L(x)=d(\{w\in L: w<x\})$ [54]. Allender and Rubinstein [7] proved that the following are equivalent:

(1) S is P-printable.

(2) S is sparse and has a ranking function computable in polynomial time.

(3) S is P-isomorphic to some tally set in P.

(4) $S\subseteq K[k\log n, n^k, \infty]$ for some constant k and $S\in\mathrm{P}$.

Note: The equivalence of (1) and (4) is due to Balcazar, and Book [8] and Hartmanis and Hemachandra [57].

6.5. *Time-bounded Kolmogorov complexity and language compression*

If A is a recursive set and x is lexicographically the ith element in A, then we know

$K(x) \leqslant \log i + c_A$ for some constant c_A not depending on x. Here we use plain Kolmogorov complexity as in the introduction of Section 2.

NOTATION. In this section let us write $K^t(x|y)$ to denote the conditional t-time-bounded Kolmogorov complexity of x, given y. Define the unconditional complexity of x as $K^t(x) = K^t(x|\varepsilon)$.

Further let $A \in \mathrm{P}$, where P is the class of problems decidable by deterministic Turing machines in polynomial time. It is seductive to think the following:

CONJECTURE. $\exists c \forall s \in A^n [K^p(s) \leqslant \log d(A^n) + c_A]$, *where* A^n *is the set of elements in* A *of length* n, *and* p *is a polynomial.*

However in polynomial time a Turing machine cannot search through 2^n strings as is assumed with A a recursive set as above. Whether or not the above conjecture is true is still an important open problem in time-bounded Kolmogorov complexity which we deal with exclusively in this section. It also has important consequences in language compression.

DEFINITION (*Goldberg and Sipser* [51]). (1) A function $f: \Sigma^* \to \Sigma^*$ is a *compression* of language L if f is one-to-one on L and, for all except finitely many $x \in L$, $|f(x)| < |x|$.

(2) A language L is *compressible in time* T if there is a compression function f for L which can be computed in time T, and also the inverse f^{-1} of f with domain $f(L)$, such that for any $x \in L$, $f^{-1}(f(x)) = x$, can be computed in time T.

(3) Compression function f *optimally compresses* a language L if, for any $x \in L$ of length n,

$$|f(x)| \leqslant \left\lceil \log\left(\sum_{i=0}^{n} d(L^i) \right) \right\rceil.$$

(4) One natural and optimal compression is *ranking*. The ranking function $r_L: L \to N$ maps $x \in L$ to its index in a lexicographical ordering of L.

Obviously, language compression is closely related to the Kolmogorov complexity of the elements in the language. *Efficient* language compression is closely related to the *time-bounded* Kolmogorov complexity of the elements of the language. By using a ranking function, we can obtain the optimal Kolmogorov complexity of any element in a recursive set, and hence, optimally compress the total recursive set. That was trivial. Our purpose is studying the polynomial-time setting of the problem. This is far from trivial.

6.5.1. *Language compression with the help of an oracle*

Let $\psi_1, \psi_2, \ldots$ be an effective enumeration of partial recursive predicates. Let T_ψ be a multitape Turing machine which computes ψ. $T_\psi(x)$ outputs 0 or 1. If T_ψ accepts x in t steps (time), then we also write $\psi^t(x) = 1$.

Definition (*Sipser*). Let x, y, p be strings in $\{0,1\}^*$. Fixing ψ, we can define KD^t_ψ of x, *conditional to* ψ and y, by

$$KD^t_\psi(x|y) = \min\{|p| : \forall v, \psi^t(v,p,y)=1 \text{ iff } v=x\},$$

and $KD^t_\psi(x|y)=\infty$ if there are no such p.

Remark. One can prove an invariance theorem similar to that of the K^t version; we can hence drop the index ψ in KD^t_ψ.

The intuition of the above definition is that while $K^t(x)$ is the length of the shortest program *generating* x in $t(|x|)$ time, $KD^t(x)$ is the length of the shortest program *accepting* only x in $t(|x|)$ time. In pure Kolmogorov complexity, these two measures differ by only an additive constant. In the resource-bounded Kolmogorov complexity, they appear quite different. The KD version appears to be somewhat simpler, and the following were proved by Sipser [146]. Let p, q be polynomials, c be a constant, and NP be an NP-complete oracle.

(1) $\forall p \exists q[KD^q(s) \leqslant K^p(s) + O(1)]$.

(2) $\forall p \exists q[K^q(s | NP) \leqslant KD^p(s) + O(1)]$.

(3) $\forall c \exists d$, if $A \subseteq \Sigma^n$ and A is accepted by a circuit of size n^c, then for each $s \in A$,

$$KD^d(s | A, i_A) \leqslant \log d(A) + \log\log d(A) + O(1),$$

where i_A depends on A and has length about $n \log d(A)$.

(4) $\forall c \exists d$, if $A \subseteq \Sigma^n$ is accepted by a circuit of size n^c and there is a string i_A such that for each $s \in A$,

$$K^d(s | A, i_A, NP) \leqslant \log d(A) + \log\log d(A) + O(1),$$

then

$$K^d(s | A, \Sigma_2) \leqslant \log d(A) + \log\log d(A) + O(1).$$

In order to prove the above results, Sipser needed an important coding lemma which will be proved again below using Kolmogorov complexity. Let $A \subseteq \Sigma^n$, $k = d(A)$ and $m = 1 + \lceil \log k \rceil$. Let $h : \Sigma^n \to \Sigma^m$ be a linear transformation given by a randomly chosen $m \times n$ binary matrix $R = \{r_{ij}\}$, i.e. for $x \in \Sigma^n$, Rx is a string $y \in \Sigma^m$ where $y_i = (\sum_j r_{ij} \times x_j) \bmod 2$. Let H be a collection of such functions. Let $A, B \subseteq \Sigma^n$ and $x \in \Sigma^n$. *h separates x within A* if for every $y \in A$, different from x, $h(y) \neq h(x)$. *h separates B within A* if it separates each $x \in B$ within A. *H separates B within A* if for each $x \in B$ some $h \in H$ separates x within A. In order to give each element in A a (logarithmic) short code, we randomly hash elements of A into short codes. If collision can be avoided, then elements of A can be described by short programs.

Coding Lemma (Sipser). *Let $A \subseteq \Sigma^n$, where $d(A)=k$. Let $m = 1 + \lceil \log k \rceil$. There is a collection H of m linear transformations $\Sigma^n \to \Sigma^m$ such that H separates A within A.*

Proof. We give the main idea ignoring relevant issues as self-delimiting descriptions, etc. Fix a random string s of length nm^2 such that $K(s|A) \geqslant |s|$. Cut x into m equal pieces.

Use the nm bits from each piece to form an $n \times m$ binary matrix in the obvious way. Thus we have constructed a set H of m random matrices. We claim that H separates A within A.

Assume this is not true. That is, for some $x \in A$, no $h \in H$ separates x within A. Hence there exist $y_1, \ldots, y_m \in A$ such that $h_i(x) = h_i(y_i)$. Hence $h_i(x - y_i) = 0$. Since $x - y_i \neq 0$, the first column of h_i corresponding to a 1 in $x - y_i$ can be expressed by the rest of the columns using $x - y_i$. Now we can describe s using the following:

- index of x in A, using $\lceil \log k \rceil$ bits,
- indices of $y_1, \ldots, y_m$, in at most $m \lceil \log k \rceil$ bits,
- matrices $h_1, \ldots, h_m$ each minus the redundant column, in $m^2n - m^2$ bits.

From the above information, given A, a short program will reconstruct h_i by the rest columns of h_i and x, y_i. The total length is only

$$m^2n - m(\log k + 1) + \log k + m(\log k) \leqslant nm^2 - 1.$$

Hence, $K(s|A) < |s|$, a contradiction. $\square$

From this lemma, Sipser also proved $\mathrm{BPP} \subseteq \Sigma_4 \cap \Pi_4$. Gács improved this to $\mathrm{BPP} \subseteq \Sigma_2 \cap \Pi_2$. We provide Gács' proof: Let $B \in \mathrm{BPP}$ be accepted by a probabilistic algorithm with error probability at most 2^{-n} on inputs of length n, which uses $m = n^k$ random bits. Let $E_x \subset \Sigma^m$ be the collection of random inputs on which M rejects x. For $x \in B$, $|E_x| \leqslant 2^{m-n}$. Letting $l = 1 + m - n$, the Coding Lemma states that there is a collection H of l linear transformations from Σ^m to Σ^l separating E_x within E_x. If x is not in B, $|E_x| > 2^{m-1}$ and by the pidgeon hole principle, no such collection exists. Hence $x \in B$ iff such an H exists. The latter can be expressed as

$$\exists H\, \forall e \in E_x\, \exists h \in H\, \forall e' \in E_x\, [e \neq e' \text{ implies } h(e) \neq h(e')].$$

The second existential quantifier has polynomial range, hence can be eliminated. Hence $\mathrm{BPP} \subseteq \Sigma_2$. Since BPP is closed under complement, $\mathrm{BPP} \subseteq \Pi_2$. Hence $\mathrm{BPP} \subseteq \Sigma_2 \cap \Pi_2$.

6.5.2. *Language compression without oracle*

Without the help of oracles, Sipser and Goldberg [51] obtained much weaker and more difficult results. For a given language L, define the density of L to be $\mu_L = \max\{\mu_L(n)\}$, where $\mu_L(n) = d(L^n)/2^n$. Goldberg and Sipser proved that if $L \in \mathrm{P}$, $k > 3$, and $\mu_L \leqslant n^{-k}$, then L can be compressed in probabilistic polynomial time; the compression function f maps strings of length n to strings of length $n - (k-3)\log n + c$ with probability approaching 1.

The above result is weak in two senses:

(1) If a language L is very sparse, say $\mu_L \leqslant 2^{-n/2}$, then one expects to compress $\frac{1}{2}n$ bits instead of only $O(\log n)$ bits given by the theorem; can this be improved?

(2) The current compression algorithm is probabilistic; can this be made deterministic?

In computational complexity, oracles sometimes help us to understand the possibility of proving a new theorem. Goldberg and Sipser show that when S, the language to be compressed, does not need to be in P and the membership query of S is given by an oracle, then the above result is optimal. Specifically, we have the following:

(1) There is a sparse language S which cannot be compressed by more than $O(\log n)$ bits by a probabilistic polynomial-time machine with an oracle for S.

(2) There is a language S, of density $\mu_S < 2^{-n/2}$, which cannot be compressed by any deterministic polynomial-time machine that uses the oracle for S.

See [151] for practical data compression techniques.

6.5.3. Ranking: optimally compressible languages

Ranking is a special and optimal case of compression. The ranking function r_L maps the strings in L to their indices in the lexicographical ordering of L. If $r_L: L \rightarrow N$ is polynomial-time computable, then so is $r_L^{-1}: N \rightarrow L$ in terms of the length of its output. We are only interested in polynomial-time computable ranking functions. In fact, there are natural language classes that are easy to compress. Goldberg and Sipser [54], and Allender [4] show that if a language L is accepted by a one-way logspace Turing machine, then r_L can be computed in polynomial time. Goldberg and Sipser also prove by diagonalization that

(a) there is an exponential-time language that cannot be compressed in deterministic polynomial time; and

(b) there is a double exponential-time language that cannot be compressed in probabilistic polynomial time.

Call C P-rankable if for all $L \in C$, r_L is polynomial-time computable. Hemachandra in [61] proved that P is P-rankable iff NP is P-rankable, and P is P-rankable iff $\mathrm{P} = \mathrm{P}^{\#\mathrm{P}}$, and PSPACE is P-rankable iff P = PSPACE. Say a set A is *k-enumeratively-rankable* if there is a polynomial-time computable function f so that for every x, $f(x)$ prints a set of k numbers, one of which is the rank of x with respect to A. Cai and Hemachandra [21] proved $\mathrm{P} = \mathrm{P}^{\#\mathrm{P}}$ iff each set $A \in \mathrm{P}$ for some k_A has a k_A-enumerative-ranker.

6.6. A Kolmogorov random reduction

The original ideas of this section belong to U. Vazirani and V. Vazirani [159]. We reformulate their results in 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 under Turing reductions. In [159] the notion is refined as follows.

Definition. A is *PR-reducible* to B, denoted by $A \leqslant_{\mathrm{PR}} B$, iff there is 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 $\Pr(T(x)$ not in $B) \geqslant \delta$.

A problem is *PR-complete* if every NP problem can be PR-reduced to it.

Vazirani and Vazirani obtained a nonnumber-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 at 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 follows.

ENCODING BY FST

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.)

The reduction is now as follows: any instance (x, y, k) of ENCODING BY FST is transformed to (xr, yr, k) for ENCODING BY TM, where $K(r|x, y) \geqslant |r| - c$ and $r \in \{\alpha, \beta\}^*$. For $\delta = 2^{-c}$, Pr(generate such an $r) \geqslant 1 - \delta$. Clearly, if there is an FST F of at most k states that outputs y on input x, then we can construct a TM which outputs yr on input xr by simply adding two new transitions from each state back to itself on α, β and outputing what it reads. If there is no such FST, then the k-state TM must reverse its read head on prefix x, or halt, when producing y. Hence it produces r without seeing r (notice the real-time requirement). Hence $K(r|x, y) = O(1)$, a contradiction. $\square$

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 starting the golden spring of Kolmogorov complexity.

Acknowledgment

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, R. Paturi, and J. Seiferas and Y. Yesha kindly supplied to us (and gave us permission to use) their unpublished material about the prime number theorem, lower bounds for probabilistic machines and Boolean matrix rank respectively. O. Watanabe and L. Longpré supplied to us interesting material in connection with the structure of complexity classes. A.Kh. Shen' and A. Verashagin translated an initial draft version of this article into Russian, and pointed out several errors. Comments of Charles Bennett, Peter van Emde Boas, Jan Heering, Evangelos Kranakis, Ker-I Ko, Danny Krizanc, Michiel van Lambalgen, Lambert Meertens, John Tromp, Umesh Vazirani, and Mati Wax 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.

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, Army Research Office Grant DAAL03-86-K-0171, and NSERC Operating Grant OGP0036747. His current affiliation is: Department of Computer Science, University of Waterloo, Ontario, Canada N2L 3G1.

Preliminary versions of parts of this article appeared as: "Two decades of applied Kolmogorov complexity; In memoriam A.N. Kolmogorov 1903–1987", in: *Proc. 3rd IEEE Structure in Complexity Theory Conference* (1988) 80–102; and: "Kolmogorovskaya swozhnost': dvadsat' let spustia", *Uspekhi Mat. Nauk* **43**(6) (1988) 128–166 (in Russian) (= *Russian Mathematical Surveys*).

References

[1] Aanderaa, S.O., On k-tape versus $(k-1)$-tape real-time computation, in: R.M. Karp, ed., *Complexity of Computation* (Amer. Mathematical Soc., Providence, RI, 1974) 75–96.

[2] Adleman, L., Time, space, and randomness, Report MIT/LCS/79/TM-131, Laboratory for Computer Science, Massachusetts Institute of Technology, 1979.

[3] Aleksandrov, P.S., A few words on A.N. Kolmogorov, *Russian Math. Surveys* **38** (1983) 5–7.

[4] Allender, E., Invertible functions, Ph.D. Thesis, Georgia Institute of Technology, Atlanta 1985.

[5] Allender, E., Some consequences of the existence of pseudorandom generators, *J. Comput. System Sci.* **39** (1989) 101–124.

[6] Allender, E. and O. Watanabe, Kolmogorov complexity and degrees of tally sets, in: *Proc. 3rd Ann. Conf. IEEE on Structure in Complexity Theory* (1988) 102–111.

[7] Allender, E.A. and R.S. Rubinstein, P-printable sets, *SIAM J. Comput.* **17** (1988) 1193–1202.

[8] Balcazar, J. and R. Book, On generalized Kolmogorov complexity, in: A.L. Selman, ed., *Structure in Complexity Theory*, Lecture Notes in Computer Science, Vol. 223 (Springer, Berlin, 1986) 334–340.

[9] 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** (1968) 1251–1254.

[10] Beame, P., Limits on the power of concurrent-write parallel machines, *Inform. and Comput.* **76** (1988) 13–28.

[11] Bennett, C.H., On random and hard to describe numbers, Mathematics Tech. Report RC 7483 (#32272), IBM T.J. Watson Research Center, Yorktown Heights, 1979; also in: M. Gardner, Mathematical Games, *Scientific American* (November 1979) 20–34.

[12] Bennett, C.H., *On the logical "depth" of sequences and their reducibilities to random sequences*, Unpublished manuscript, IBM T.J. Watson Research Center, Yorktown Heights (1981/2).

[13] Bennett, C.H., Dissipation, information, computational complexity and the definition of organization, in: D. Pines, ed., *Emerging Syntheses in Science* (*Proc. Founding Workshops of the Santa Fe Institute, 1985*) (Addison-Wesley, Reading, MA, 1987) 297–313.

[14] Bennett, C.H., Logical depth and physical complexity, in: R. Herken, ed., *The Universal Turing Machine; A Half-Century Survey* (Oxford University Press, Oxford and Kammerer & Unverzagt, Hamburg, 1988) 227–258.

[15] Berman, L. and J. Hartmanis, On isomorphisms and density of NP and other complete sets, *SIAM J. Comput.* **6** (1977) 305–327.

[16] Blumer, A., A. Ehrenfeucht, D. Haussler and M. Warmuth, Classifying learnable geometric concepts with the Vapnik–Chervonenkis dimension, in: *Proc. 18th Ann. ACM Symp. on Theory of Computing* (1986) 273–282.

[17] Bogolyubov, N.N., B.V. Gnedenko and S.L. Sobolev, Andrei Nikolaevich Kolmogorov (on his eightieth birthday), *Russian Math. Surveys* **38** (1983) 9–27.

[18] Book, R., P. Orponen, D. Russo and O. Watanabe, Lowness properties of sets in the exponential-time hierarchy, *SIAM J. Comput.* **17** (1988) 504–516.

[19] 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, in: *Proc. 20th Ann. IEEE Symp. on Foundations of Computer Science* (1979) 319–328.

[20] Borodin, A. and S. Cook, A time-space tradeoff for sorting on a general sequential model of computation, in: *Proc. 12th Ann. ACM Symp. on Theory of Computing* (1980) 294–301.

[21] Cai, J. and L. Hemachandra, Enumerative counting is hard, *Inform. and Comput.* **82** (1989) 34–44.

[22] Carnap, R., *Logical Foundations of Probability* (Univ. of Chicago Press, Chicago, IL, 1950).

[23] Carter, J. and M. Wegman, Universal classes of hashing functions, *J. Comput. System Sci.* **18** (1979) 143–154.

[24] Chaitin, G.J., On the length of programs for computing finite binary sequences, *J. Assoc. Comput. Mach.* **13** (1966) 547–569.

[25] Chaitin, G.J., On the length of programs for computing finite binary sequences: statistical considerations, *J. Assoc. Comput. Mach.* **16** (1969) 145–159.

[26] Chaitin, G.J., Information-theoretic limitations of formal systems, *J. Assoc. Comput. Mach.* **21** (1974) 403–424.

[27] Chaitin, G.J., Randomness and mathematical proof, *Scientific American* **232** (May 1975) 47–52.

[28] Chaitin, G.J., A theory of program size formally identical to information theory, *J. Assoc. Comput. Mach.* **22** (1975) 329–340.

[29] Chaitin, G.J., Information-theoretic characterizations of recursive infinite strings, *Theoret. Comput. Sci.* **2** (1976) 45–48.

[30] Chaitin, G.J., Algorithmic information theory, *IBM J. Res. Develop.* **21** (1977) 350–359.

[31] Chaitin, G.J., Toward a mathematical definition of "life", in: M. Tribus, ed., *The Maximal Entropy Formalism* (MIT Press, Cambridge, MA, 1979) 477–498.

[32] Chaitin, G.J., Gödel's theorem and information, *Internat. J. Theoret. Phys.* **22** (1982) 941–954; reprinted in: T. Tymoczko, ed., *New Directions in the Philosophy of Mathematics* (Birkhäuser, Boston, 1986).

[33] Chaitin, G.J., *Algorithmic Information Theory* (Cambridge Univ. Press, Cambridge, 1987).

[34] Champernowne, D.G., The construction of decimals normal in the scale of ten, *J. London Math. Soc. (2)* **8** (1933) 254–260.

[35] Chrobak, M., Hierarches of one-way multihead automata languages, *Theoret. Comput. Sci.* **48** (1986) 153–181.

[36] Chrobak, M. and M. Li, $k+1$ heads are better than k for PDAs, *J. Comput. System Sci.* **37** (1988) 144–155.

[37] Church, A., On the concept of a random sequence, *Bull. Amer. Math. Soc.* **46** (1940) 130–135.

[38] Cover, T.M., Universal gambling schemes and the complexity measures of Kolmogorov and Chaitin, Tech. Report 12, Statistics Dept., Stanford Univ., Palo Alto, CA, 1974.

[39] Cuykendall, R.R., Kolmogorov information and VLSI lower bounds, Ph.D. Thesis, Univ. of California, Los Angeles, CA, 1984.

[40] Daley, R.P., On the inference of optimal descriptions, *Theoret. Comput. Sci.* **4** (1977) 301–309.

[41] Dietzfelbinger, M., Lower bounds on computation time for various models in computational complexity theory, Ph.D. Thesis, Dept. of Computer Science, Univ. of Illinois at Chicago, 1987.

[42] Ďuriš, P., Z. Galil, W. Paul and R. Reischuk, Two nonlinear lower bounds for online computations, *Inform. and Control* **60** (1984) 1–11.

[43] Erdős, P. and J. Spencer, *Probabilistic Methods in Combinatorics* (Academic Press, New York, 1974).

[44] Floyd, R., Review 14, *Comput. Rev.* **9** (1968) 280.

[45] Gács, P., On the symmetry of algorithmic information, *Soviet Math. Dokl.* **15** (1974) 1477–1480; Correction, *Ibidem* **15** (1974) 1480.

[46] Gács, P., Randomness and probability—complexity of description, in: Kotz-Johnson, ed., *Encyclopedia of Statistical Sciences* (Wiley, New York, 1986) 551–555.

[47] Gács, P., Lecture notes on descriptional complexity and randomness, Unpublished manuscript, Boston University, Boston, MA, 1987.

[48] Galil, Z. and J. Seiferas, Time-space optimal matching, in: *Proc. 13th Ann. ACM Symp. on Theory of Computing* (1981) 106–113.

[49] Galil, Z., R. Kannan, and E. Szemeredi, On nontrivial separators for *k*-page graphs and simulations by nondeterministic one-tape Turing machines, *J. Comput. System Sci.* **38** (1989) 134–149.

[50] Gallaire, H., Recognition time of context-free languages by on-line Turing machines, *Inform. and Control* **15** (1969) 288–295.

[51] Gao, Q. and M. Li, An application of minimum description length principle to online recognition of handprinted alphanumerals, in: *Proc. 11th Internat. Joint Conf. on Artificial Intelligence*, Detroit, MI (1989).

[52] Gnedenko, B.V., Andrei Nikolaevich Kolmogorov (on the occasion of his seventieth birthday), *Russian Math. Surveys* **28** (1973) 5–16.

[53] Gold, E.M., Language identification in the limit, *Inform. and Control* **10** (1967) 447–474.

[54] Goldberg, Y. and M. Sipser, Compression and ranking, *SIAM J. Comput.* **19** (1990).

[55] Harrison, M.A. and O.H. Ibarra, Multi-head and multi-tape pushdown automata, *Inform. and Control* **13** (1968) 433–470.

[56] Hartmanis, J., Generalized Kolmogorov complexity and the structure of feasible computations, in: *Proc. 24th Ann. IEEE Symp. on Foundations of Computer Science* (1983) 439–445.

[57] Hartmanis, J. and L. Hemachandra, On sparse oracles separating feasible complexity classes, in: *Proc. 3rd Symp. on Theoretical Aspects of Computer Science* (*STACS '86*), Lecture Notes in Computer Science, Vol. 210 (Springer, Berlin, 1986) 321–333.

[58] Haussler, D., N. Littlestone and M. Warmuth, Expected mistake bounds for on-line learning algorithms, Manuscript, 1988.

[59] Heide, F. Meyer auf der and A. Wigderson, The complexity of parallel sorting, in: *Proc. 17th Ann. ACM Symp. on Theory of Computing* (1985) 532–540.

[60] Hemachandra, L., Can P and NP manufacture randomness?, Tech. Report TR86-795, Dept. of Computer Science, Cornell Univ., Ithaca, NY, 1986.

[61] Hemachandra, L., On ranking, in: *Proc. 2nd Ann. IEEE Conf. on Structure in Complexity Theory* (1987) 103–117.

[62] Hopcroft, J.E. and J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation* (Addison-Wesley, Reading, MA, 1979).

[63] Huynh, D.T., Non-uniform complexity and the randomness of certain complete languages, Tech. Report TR 85-34, Iowa State Univ., 1985.

[64] Huynh, D.T., Resource-bounded Kolmogorov complexity of hard languages, in: *Structure in Complexity Theory*, Lecture Notes in Computer Science, Vol. 223 (Springer, Berlin, 1986) 184–195.

[65] Ibarra, O.H. and C.E. Kim, On 3-head versus 2-head finite automata, *Acta Inform.* **4** (1975) 193–200.

[66] Israeli, A. and S. Moran, Private communication.

[67] Jeffreys, Z., *Theory of Probability* (Oxford at the Clarendon Press, Oxford, 3rd ed., 1961).

[68] Kearns, M., M. Li, L. Pitt and L. Valiant, On the learnability of Boolean formulae, in: *Proc. 19th Ann. ACM Symp. on Theory of Computing* (1987) 285–295.

[69] Kearns, M., M. Li, L. Pitt and L. Valiant, Recent results on Boolean concept learning, in: T. Mitchell, ed., *Proc. 4th Workshop on Machine Learning*, (Morgan Kaufmann, Los Altos, CA, 1987) 337–352.

[70] Kemeny, J.G., The use of simplicity in induction, *Philos. Rev.* **62** (1953) 391–408.

[71] Knuth, D., *Semi-numerical Algorithms* (Addison-Wesley, Reading, MA, 2nd ed., 1981).

[72] Knuth, D.E., Big Omicron and big Omega and big Theta, *SIGACT News* **8** (2) (1976) 18–24.

[73] Ko, K.-I, Resource-bounded program-size complexity and pseudorandom sequences, Tech. Report, Dept. of Computer Science, Univ. of Houston, Houston, TX, 1983.

[74] Kolmogorov, A.N., *Grundbegriffe der Wahrscheinlichkeitsrechnung* (Springer, Berlin, 1933); *Osnovnye Poniatija Teorii Verojatnostej* (Nauka, Moscow, 2nd Russian ed., 1974).

[75] KOLMOGOROV, A.N., On tables of random numbers, *Sankhyā Ser. A* **25** (1963) 369–376.

[76] KOLMOGOROV, A.N., Three approaches to the quantitative definition of information, *Problems Inform. Transmission* **1** (1) (1965) 1–7.

[77] KOLMOGOROV, A.N., Logical basis for information theory and probability theory, *IEEE Trans. on Inform. Theory* **14** (5) (1968) 662–664.

[78] KOLMOGOROV, A.N., Combinatorial foundations of information theory and the calculus of probabilities, *Russian Math. Surveys* **38** (1983) 29–40.

[79] KOLMOGOROV, A.N., *Information Theory and Theory of Algorithms, Selected Works Vol. 3* (Nauka, Moscow, 1987) (in Russian).

[80] KOLMOGOROV, A.N. and V.A. USPENSKII, Algorithms and randomness, *SIAM J. Theory Probab. Appl.* **32** (1987) 389–412.

[81] LAMBALGEN, M. VAN, Von Mises' definition of random sequences reconsidered, *J. Symbolic Logic* **52** (1987) 725–755.

[82] LAMBALGEN, M. VAN, Random sequences, Ph.D. Thesis, Faculty of Mathematics and Computer Science, Univ. van Amsterdam, Amsterdam, 1987.

[83] LAPLACE, P.S., *A Philosophical Essay on Probabilities* (Dover, New York; original publication, 1819).

[84] LEVIN, L.A., Universal search problems, *Problems Inform. Transmission* **9** (1973) 265–266.

[85] LEVIN, L.A., On the notion of a random sequence, *Soviet Math. Dokl.* **14** (1973) 1413–1416.

[86] LEVIN, L.A., Laws of information conservation (non-growth) and aspects of the foundation of probability theory, *Problems Inform. Transmission* **10** (1974) 206–210.

[87] LEVIN, L.A., Do chips need wires?, Manuscript/NSF proposal MCS-8304498, Computer Science Dept., Boston Univ., 1983.

[88] LEVIN, L.A., Randomness conservation inequalities; information and independence in mathematical theories, *Inform. and Control* **61** (1984) 15–37.

[89] LI, M., Simulating two pushdowns by one tape in $O(n^{1.5}(\log n)^{0.5})$ time, *J. Comput. System Sci.* **37** (1988) 101–116.

[90] LI, M., Lower bounds in computational complexity, Ph.D. Thesis, Report TR-85-663, Computer Science Dept., Cornell Univ., Ithaca, NY, 1985.

[91] LI, M., L. LONGPRÉ and P.M.B. VITÁNYI, On the power of the queue, in: *Structure in Complexity Theory*, Lecture Notes in Computer Science, Vol. 223 (Springer, Berlin, 1986) 219–233.

[92] LI, M. and Y. YESHA, String-matching cannot be done by 2-head 1-way deterministic finite automata, *Inform. Process. Lett.* **22** (1986) 231–235.

[93] LI, M. and Y. YESHA, New lower bounds for parallel computation, *J. Assoc. Comput. Mach.* **36** (1989) 671–680.

[94] LI, M. and P.M.B. VITÁNYI, Tape versus queue and stacks: the lower bounds, *Inform. and Comput.* **78** (1988) 56–85.

[95] LI, M. and P.M.B. VITÁNYI, A new approach to formal language theory by Kolmogorov complexity, in: *Proc. 16th Internat. Coll. on Automata, Languages and Programming*, Lecture Notes in Computer Science, Vol. 372 (Springer, Berlin, 1989) 506–520.

[96] LI, M. and P.M.B. VITÁNYI, Inductive reasoning and Kolmogorov complexity, in: *Proc. 4th Ann. IEEE Structure in Complexity Theory Conf.* (1989) 165–185.

[97] LI, M. and P.M.B. VITÁNYI, *An Introduction to Kolmogorov Complexity and Its Applications* (Addison-Wesley, Reading, MA, to appear).

[98] LIPTON, R. and R. SEDGEWICK, Lower bounds for VLSI, in: *Proc. 13th Ann. ACM Symp. on Theory of Computing* (1981) 300–307.

[99] LITTLESTONE, N., Learning quickly when irrelevant attributes abound: a new linear threshold algorithm, in: *Proc. 28th Ann. IEEE Symp. on Foundations of Computer Science* (1987) 68–77.

[100] LITTLEWOOD, J.E., The dilemma of probability theory, in: B. Bollobás, ed., *Littlewood's Miscellany* (Cambridge Univ. Press, Cambridge, revised ed., 1986) 71–73.

[101] LONGPRÉ, L., Resource bounded Kolmogorov complexity, a link between computational complexity and information theory, Ph.D. Thesis, Tech. Report TR-86-776, Computer Science Dept., Cornell Univ., Ithaca, NY, 1986.

[102] LOUI, M., Optimal dynamic embedding of trees into arrays, *SIAM J. Comput.* **12** (1983) 463–472.

[103] LOVELAND, D.W., On minimal-program complexity measures, in: *Proc. ACM Symp. on Theory of Computing* (1969) 61–65.

[104] LOVELAND, D.W., A variant of the Kolmogorov concept of complexity, *Inform. and Control* **15** (1969) 510–526.

[105] MAASS, W., Combinatorial lower bound arguments for deterministic and nondeterministic Turing machines, *Trans. Amer. Math. Soc.* **292** (1985) 675–693.

[106] MAASS, W. and G. SCHNITGER, An optimal lower bound for Turing machines with one work tape and a two-way input tape, in: *Structure in Complexity Theory*, Lecture Notes in Computer Science, Vol. 223 (Springer, Berlin, 1986) 249–264.

[107] MAASS, W., G. SCHNITGER and E. SZEMEREDI, Two tapes are better than one for off-line Turing machines, in: *Proc. 19th Ann. ACM Symp. on Theory of Computing* (1987) 94–100.

[108] MAIRSON, H.G., The program complexity of searching a table, in: *Proc. 24th IEEE Symp. on Foundations of Computer Science* (1983) 40–47.

[109] MARTIN-LÖF, P., The definition of random sequences, *Inform. and Control* **9** (1966) 602–619.

[110] MARTIN-LÖF, P., Algorithmen und zufällige Folgen, Lecture notes, Univ. of Erlangen, 1966.

[111] MARTIN-LÖF, P., Complexity oscillations in infinite binary sequences, *Z. Wahrsch. Verw. Gebiete* **19** (1971) 225–230.

[112] MEHLHORN, K., On the program-size of perfect and universal hash functions, in: *Proc. 23rd Ann. IEEE Symp. on Foundations of Computer Science* (1982) 170–175.

[113] 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: A.H. Traub, ed., *John von Neumann, Collected Works, Vol. V* (MacMillan, New York, 1963).

[114] MINSKY, M.L., Steps towards Artificial Intelligence, *Proc. I.R.E.* (January 1961) 8–30.

[115] MISES, R. VON, *Probability, Statistics and Truth* (MacMillan, New York, 1939; reprint, Dover, New York, 1981).

[116] MIYANO, S., A hierarchy theorem for multihead stack-counter automata, *Acta Inform.* **17** (1982) 63–67.

[117] MIYANO, S., Remarks on multihead pushdown automata and multihead stack automata, *J. Comput. System Sci.* **27** (1983) 116–124.

[118] NATARAJAN, B.K., Personal communication, 1988.

[119] NELSON, C.G., One-way automata on bounded languages, Tech. Report TR14-76, Aiken Computer Lab., Harvard Univ., 1976.

[120] NEUMANN, J. VON, Various techniques used in connection with random digits, in: A.H. Traub, ed., *John von Neumann, Collected Works, Vol. V* (MacMillan, New York, 1963).

[121] Obituary, Mr. Andrei Kolmogorov—giant of mathematics, *Times* (October 26, 1987).

[122] PARBERRY, I., A complexity theory of parallel computation, Ph.D. Thesis, Dept. of Computer Science, Warwick Univ., Coventry, UK, 1984.

[123] PATURI, R. and J. SIMON, Lower bounds on the time of probabilistic on-line simulations, in: *Proc. 24th Ann. IEEE Symp. on Foundations of Computer Science* (1983) 343–350.

[124] PAUL, W., Kolmogorov's complexity and lower bounds, in: L. Budach, ed., *Proc. 2nd Internat. Conf. on Fundamentals of Computation Theory* (Akademie Verlag, Berlin, 1979) 325–334.

[125] PAUL, W., On heads versus tapes, *Theoret. Comput. Sci.* **28** (1984) 1–12.

[126] PAUL, W.J., J.I. SEIFERAS and J. SIMON, An information theoretic approach to time bounds for on-line computation, *J. Comput. System Sci.* **23** (1981) 108–126.

[127] PAUL, W.J., On-line simulation of $k+1$ tapes by k tapes requires nonlinear time, *Inform. and Control* (1982) 1-8.

[128] PETERSON, G., Succinct representations, random strings and complexity classes, in: *Proc. 21st Ann. IEEE Symp. on Foundations of Computer Science* (1980) 86–95.

[129] POPPER, K.R., *The Logic of Scientific Discovery* (Univ. of Toronto Press, Toronto, 1959).

[130] QUINLAN, J. and R. RIVEST, Inferring decision trees using the minimum description length principle, *Inform. and Comput.* **80** (1989) 227–248.

[131] REISCH, S. and G. SCHNITGER, Three applications of Kolmogorov-complexity, in: *Proc. 23rd Ann. IEEE Symp. on Foundations of Computer Science* (1982) 45–52.

[132] RISSANEN, J., Modeling by the shortest data description, *Automatica—J. IFAC* **14** (1978) 465–471.

[133] RISSANEN, J., A universal prior for integers and estimation by minimum description length, *Ann. Statist.* **11** (1982) 416–431.

[134] RIVEST, R., Learning decision-lists, Unpublished manuscript, Lab. for Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 1986.

[135] ROGERS JR., H., *Theory of Recursive Functions and Effective Computability* (McGraw Hill, New York, 1967).

[136] ROSENBERG, A., Nonwriting extensions of finite automata, Ph.D. Thesis, Aiken Computer Lab., Harvard Univ., Cambridge, MA, 1965.

[137] ROSENBERG, A., On multihead finite automata, *IBM J. Res. Develop.* **10** (1966) 388–394.

[138] SCHNORR, C.P., Eine Bemerkung zum Begriff der zufälligen Folge, *Z. Wahrsch. Verw. Gebiete* **14** (1969/70) 27–35.

[139] SCHNORR, C.P., *Zufälligkeit und Wahrscheinlichkeit; Eine algorithmische Begründung der Wahrscheinlichkeitstheorie*, Lecture Notes in Mathematics, Vol. 218 (Springer, Berlin, 1971).

[140] SCHNORR, C.P., Process complexity and effective random tests, *J. Comput. System Sci.* **7** (1973) 376–388.

[141] SCHNORR, C.P., A survey of the theory of random sequences, in: R.E. Butts and J. Hintikka, eds., *Basic Problems in Methodology and Linguistics* (Reidel, Dordrecht, 1977) 193–210.

[142] SEIFERAS, J., The symmetry of information, and an application of the symmetry of information, Notes, Computer Science Dept, Univ. of Rochester, 1985.

[143] SEIFERAS, J., A simplified lower bound for context-free-language recognition, *Inform. and Control* **69** (1986) 255–260.

[144] SHANNON, C.E. and W. WEAVER, *The Mathematical Theory of Communication* (Univ. of Illinois Press, Urbana, IL, 1949).

[145] SHANNON, C.E., A universal Turing machine with two internal states, in: C.E. Shannon and J. McCarthy, eds., *Automata Studies* (Princeton Univ. Press, Princeton, NJ, 1956).

[146] SIPSER, M., A complexity theoretic approach to randomness, in: *Proc. 15th Ann. ACM Symp. on Theory of Computing* (1983) 330–335.

[147] SKIENA, S.S., Further evidence for randomness in π, *Complex Systems* **1** (1987) 361–366.

[148] SOLOMONOFF, R.J., A preliminary report on a general theory of inductive inference, Tech. Report ZTB-138, Zator Company, Cambridge, MA, 1960.

[149] SOLOMONOFF, R.J., A formal theory of inductive inference, Part 1 and Part 2, *Inform. and Control* **7** (1964) 1–22 and 224–254.

[150] SOLOMONOFF, R.J., Complexity-based induction systems: comparisons and convergence theorems, *IEEE Trans. Inform. Theory* **24** (1978) 422–432.

[151] STORER, J., *Data Compression: Method and Theory* (Computer Science Press, Rockville, MD, 1988).

[152] SUDBOROUGH, I.H., Computation by multi-head writing finite automata, Ph.D. Thesis, Pennsylvania State Univ., University Park, PA, 1974.

[153] SUDBOROUGH, I.H., One-way multihead writing finite automata, *Inform. and Control* **30** (1976) 1–20.

[154] THOMPSON, C.D., Area-time complexity for VLSI, in: *Proc. 11th Ann. ACM Symp. on Theory of Computing* (1979) 81–88.

[155] TURING, A.M., On computable numbers with an application to the Entscheidungsproblem, *Proc. London Math. Soc.* **42** (1936) 230–265; Correction, *Ibidem* **43** (1937) 544–546.

[156] VALIANT, L. and G. BREBNER, Universal schemes for parallel communication, in: *Proc. 13th Ann. ACM Symp. on Theory of Computing* (1981) 263–277.

[157] VALIANT, L.G., A theory of the learnable, *Comm. ACM* **27** (1984) 1134–1142.

[158] VALIANT, L.G., Deductive learning, *Philos. Trans. Royal Soc. Lond. Ser. A* **312** (1984) 441–446.

[159] VAZIRANI, U. and V. VAZIRANI, A natural encoding scheme proved probabilistic polynomial complete, *Theoret. Comput. Sci.* **24** (1983) 291–300.

[160] VILLE, J., *Etude Critique du Concept de Collectif* (Gauthier-Villars, Paris, 1939).

[161] VITÁNYI, P.M.B. and L. MEERTENS, Big Omega versus the wild functions, *SIGACT News* **16** (4) (1985) 56–59.

[162] VITÁNYI, P.M.B., On two-tape real-time computation and queues, *J. Comput. System Sci.* **29** (1984) 303–311.

[163] Vitányi, P.M.B., On the simulation of many storage heads by one, *Theoret. Comput. Sci.* **34** (1984) 157–168.

[164] Vitányi, P.M.B., Square time is optimal for the simulation of a pushdown store by an oblivious one-head tape unit, *Inform. Process. Lett.* **21** (1985) 87–91.

[165] Vitányi, P.M.B., An $N^{1.618}$ lower bound on the time to simulate one queue or two pushdown stores by one tape, *Inform. Process. Lett.* **21** (1985) 147–152.

[166] Vitányi, P.M.B., Andrei Nikolaevich Kolmogorov, *CWI Quarterly* **1** (2) (June 1988) 3–18.

[167] Wald, A., Sur la notion de collectif dans le calcul des probabilités, *C.R. Acad. Sci.* **202** (1936) 1080–1083.

[168] Watanabe, O., Comparison of polynomial time completeness notions, *Theoret. Comput. Sci.* **53** (1987) 249–265.

[169] Willis, D.G., Computational complexity and probability constructions, *J. Assoc. Comput. Mach.* **17** (1970) 241–259.

[170] Yao, A., Theory and application of trapdoor functions, in: *Proc. 23rd Ann. IEEE Symp. on Foundations of Computer Science* (1982) 80–91.

[171] Yao, A.C.-C. and R.L. Rivest, $k+1$ heads are better than k, *J. Assoc. Comput. Mach.* **25** (1978) 337–340.

[172] Yesha, Y., Time-space tradeoffs for matrix multiplication and discrete Fourier transform on any general random access computer, *J. Comput. System Sci.* **29** (1984) 183–197.

[173] 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. Surveys* **25** (1970) 83–124.

[174] Cover, T.M., P. Gács and R.M. Gray, Kolmogorov's contributions to information theory and algorithmic complexity, *Ann. Probab.* **17** (1989) 840–865.

[175] Itkis, G. and L.A. Levin, Power of fast VLSI models is insensitive to wires' thinness, in: *Proc. 30th Ann. IEEE Symp. on Foundations of Computer Science* (1989) 402–409.

[176] Li, M. and P.M.B. Vitányi, A theory of learning simple concepts under simple distributions and average case complexity for the universal distribution, in: *Proc. 30th Ann. IEEE Symp. on Foundations of Computer Science* (1989) 34–39.

[177] Wallace, C.S. and P.R. Freeman, Estimation and inference by compact coding, *J. Royal Statist. Soc. Ser. B* **49** (1987) 240–251; Discussion, *Ibidem* **49** (1987) 252–265.