2 Algorithmic Complexity

The most natural approach to defining the quantity of information is clearly to define it in relation to the individual object (be it Homer's *Odyssey* or a particular type of dodo) rather than in relation to a set of objects from which the individual object may be selected. To do so, one could define the quantity of information in an object in terms of the number of bits required to losslesly describe it. A description of an object is evidently useful in this sense only if we can reconstruct the full object from this description.

We aim at something different from C.E. Shannon's theory of communication, which deals with the specific technological problem of data transmission, that is, with the information that needs to be transmitted in order to select an object from a previously agreed-upon set of alternatives; Section 1.11. Our task is to widen the limited set of alternatives until it is universal. We aim at a notion of absolute information of individual objects, that is, the information that by itself describes the object completely.

Intuition tells us that some objects are complicated and some objects are simple. For instance, a number like 2^{1000} is certainly very simple (we have just expressed it in a few bits); yet evidently there are numbers of a thousand bits for which it is hard to see how we can find a description requiring many fewer than a thousand bits. Such hard-to-describe numbers would be their own shortest descriptions.

We require both an agreed-upon universal description method and an agreed-upon mechanism to produce the object from its alleged description. This would appear to make the information content of an object depend on whether it is particularly favored by the description method

DOI: 10.1007/978-0-387-49820-1_2, © Springer Science + Business Media, LLC 2008

we have selected. By 'favor' we mean to produce short descriptions in terms of bits.

For instance, it is well known that certain programming languages favor symbolic computations, while other programming languages favor arithmetic computations, even though all of them are universal. The notion of information content of individual objects can be useful only if the quantity of information is an attribute of the object *alone* and is independent of the means of description. It is a priori by no means obvious that this is possible. Relatively recent advances resulting in the great ideas of computability theory from the 1930s onward have made it possible to design a universal description method that appears to meet our goals.

Denote the set of objects by S, and assume some standard enumeration of objects x by natural 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 view such a method as a partial function over the nonnegative integers defined by $n = f(p)$. We do not yet assume that f is recursive, 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. With each natural number p associate the length of the finite binary string identified with p as in Equation 1.3. Denote this length by $l(p)$.

For each object x in S, the complexity of object x with respect to the specifying method f is defined as

$$C_f(x) = \min\{l(p) : f(p) = n(x)\},$$

and $C_f(x) = \infty$ if there are no such p. In computer science terminology we would say that p is a program and f a computer, so that $C_f(x)$ is the minimal length of a program for f (without additional input) to compute output x.

Considering distinct methods $f_1, f_2, \ldots, f_r$ of specifying the objects of S, it is easy to construct a new method f that assigns to each object x in S a complexity $C_f(x)$ that exceeds only by c (less than about $\log r$) the minimum of $C_{f_1}(x), C_{f_2}(x), \ldots, C_{f_r}(x)$. The only thing we have to do is to reserve the first $\log r$ bits of p to identify the method f_i that should be followed, using as a program the remaining bits of p.

We say that a method f *minorizes* a method g (additively) if there is a constant c such that for all x,

$$C_f(x) \leq C_g(x) + c.$$

Above we have shown how to construct a method f that minorizes each of the methods $f_1, \ldots, f_r$ with constant $c \approx \log r$. Two methods f and g are called *equivalent* if each of them minorizes the other.

Consider the hierarchy of equivalence classes of methods with respect to minorization. Kolmogorov has remarked that the idea of 'description length' would be useless if the constructed hierarchy did not have certain niceness properties. In particular, we would like such a hierarchy to have a unique minimal element: the equivalence class of description methods that minorize all other description methods. Some sets of description methods do have a unique minimal element, while other sets of description methods don't.

Definition 2.0.1 Let $\mathcal{C}$ be a subclass of the partial functions over the nonnegative integers. A function f is *additively optimal* (a special type of universality) for $\mathcal{C}$ if it belongs to $\mathcal{C}$ and if for every function $g \in \mathcal{C}$ there is a constant $c_{f,g}$ such that $C_f(x) \leq C_g(x) + c_{f,g}$, for all x. (Here $c_{f,g}$ depends on f and g, but not on x.) Replacing x by $\langle x, y \rangle$, with $\langle \cdot \rangle$ the standard recursive bijective pairing function, yields the definition for a class of two-variable functions.

Clearly, all additively optimal methods f, g of specifying objects in S are equivalent in the following way:

$$|C_f(x) - C_g(x)| \leq c_{f,g},$$

for all x, where $c_{f,g}$ is a constant depending only on f and g. Thus, from an asymptotic point of view, the complexity $C(x)$ of an object x, when we restrict ourselves to optimal methods of specification, does not depend on accidental peculiarities of the chosen optimal method.

Example 2.0.1 Consider the class of description methods consisting of *all* partial functions over the nonnegative integers. Every additively optimal function f for this class must be unbounded. Take an infinite sequence $x_1, x_2, \ldots$ such that $C_f(x_i) \geq i$. Define the function g by $g(i) = x_i$. Clearly, $C_g(x_i) = \log i + O(1) \ll C_f(x_i)$. Therefore, f cannot be additively optimal. Thus, there is no additively optimal partial function, and the hierarchy of complexities with respect to the partial functions does not have *any* minimal element.

The development of the theory of Kolmogorov complexity is made possible by the remarkable fact that the class of partial *recursive* functions (defined in Section 1.7) possesses a universal element that is additively optimal. Under this relatively natural restriction on the class of description methods (that is, to partial recursive functions) we obtain a well-behaved hierarchy of complexities. $\diamond$

We begin by worrying about notation. There are several variants of Kolmogorov complexity, with notations that are not used consistently among different authors or even by the same author at different times.

In the main text of this book we shall concentrate on two major variants of Kolmogorov complexity. It seems educationally the right approach to first study Kolmogorov complexity as originally defined by Solomonoff, Kolmogorov, and Chaitin because it is intuitively clearer.

Some mathematical technicalities will naturally lead up to, and justify, the less intuitive version of Kolmogorov complexity. The first type we call *plain* Kolmogorov complexity, and the second type we call *prefix* Kolmogorov complexity. We use C to denote the plain Kolmogorov complexity. We reserve K for the prefix type. Fortunately, the majority of theorems we derive for plain Kolmogorov complexity carry over unchanged and with the same proofs to the prefix version. The difference is that the prefix version is tweaked to have just the right quantitative properties for some desired uses and applications.

2.1 The Invariance Theorem

Identify an object x from a countably infinite sample space S with its index $n(x)$. Consider the class of description methods

$$\{\phi : \phi \text{ is a partial recursive function}\}.$$

Consider the particular problem of describing objects consisting of natural numbers in terms of programs consisting of finite strings of 0's and 1's. Just as in information theory, Section 1.11, where the entropy and information of a message over an alphabet of any size are expressed in the normalized format of bits, the restriction of the programs to a binary alphabet does not imply any loss of generality. In both cases, changing alphabet size leaves all statements invariant up to an appropriate logarithmic multiplicative factor related to the alphabet sizes involved; see Exercise 2.1.9.

The *invariance theorem*, Theorem 2.1.1 below, is the cornerstone for the subsequent development of the theory. In fact, for many later applications it embodies the *entire* theoretical foundation. Recall Definition 2.0.1 of a function that is additively optimal (a special type of universality) for a class of functions. We give the unconditional version as a preliminary lemma.

Lemma 2.1.1 *There is an additively optimal universal partial recursive function.*

Proof. Let ϕ_0 be the function computed by a universal Turing machine U. Machine U expects inputs of the format

$$\langle n, p \rangle = \underbrace{11\ldots1}_{l(n) \text{ times}} \; 0\, np\,.$$

The interpretation is that the total program $\langle n, p \rangle$ is a two-part code of which the first part consists of a self-delimiting encoding of T_n and the second part is the literally rendered program p. In this way, U can first parse the binary input into the T_n-part and the p-part, and subsequently simulate the computation of T_n started with program p as its input (Section 1.7). That is, $\phi_0(\langle n, p\rangle) = \phi_n(p)$. What happens if U gets the program $0p$? By convention we can set $U = T_0$ and therefore $U(0p) = U(p)$. Altogether, if T_n computes the partial recursive function ϕ_n, then

$$C_{\phi_0}(x) \le C_{\phi_n}(x) + c_{\phi_n},$$

where c_{ϕ_n} can be set to $2l(n) + 1$. □

For many applications we require a generalization to a conditional version, as follows. The difficulty of specifying an object can be facilitated when another object is already specified. We define the complexity of an object x, given an object y. Fix an effective enumeration of Turing machines $T_1, T_2, \ldots$ as in Section 1.7. The Turing machines use a tape alphabet $\{0, 1, B\}$, and the input to a Turing machine is a program consisting of a contiguous string of 0's and 1's, delimited by blanks B on both sides. In this way, a Turing machine can detect the end of its program. The effective enumeration of Turing machines induces an effective enumeration of partial recursive functions $\phi_1, \phi_2, \ldots$ such that T_i computes ϕ_i for all i. As above, $\langle\cdot\rangle : \mathcal{N} \times \mathcal{N} \to \mathcal{N}$ is a standard recursive bijective pairing function mapping the pair (x, y) to the singleton $\langle x, y\rangle$. We can iterate this as $(x, y, z) = \langle x, \langle y, z\rangle\rangle$.

Definition 2.1.1 Let x, y, p be natural numbers. Any partial recursive function ϕ, together with p and y, such that $\phi(\langle y, p\rangle) = x$, is a *description* of x. The *complexity* C_ϕ of x *conditional* to y is defined by

$$C_\phi(x|y) = \min\{l(p) : \phi(\langle y, p\rangle) = x\},$$

and $C_\phi(x|y) = \infty$ if there are no such p. We call p a *program* to compute x by ϕ, given y.

Theorem 2.1.1 *There is an additively optimal universal partial recursive function ϕ_0 for the class of partial recursive functions to compute x given y. Therefore, $C_{\phi_0}(x|y) \le C_\phi(x|y) + c_\phi$ for all partial recursive functions ϕ and all x and y, where c_ϕ is a constant depending on ϕ but not on x or y.*

Proof. Let ϕ_0 be the function computed by a universal Turing machine U such that U started on input $\langle y, \langle n, p\rangle\rangle$ simulates T_n on input $\langle y, p\rangle$ (Section 1.7). That is, if T_n computes the partial recursive function ϕ_n, then $\phi_0(\langle y, \langle n, p\rangle\rangle) = \phi_n(\langle y, p\rangle)$. Hence, for all n,

$$C_{\phi_0}(x|y) \le C_{\phi_n}(x|y) + c_{\phi_n},$$

where $c_{\phi_n} = 2l(n) + 1$. □

The key point is not that the universal description method necessarily gives the shortest description in each case, but that no other description method can improve on it infinitely often by more than a fixed constant. Note also that the optimal complexity $C_{\phi_0}(x|y)$ is defined for all x and y. Namely, for each x and y we can find a Turing machine that computes output x, given y, for some input p (such as the Turing machine that outputs x for all inputs).

For *every* pair ψ, ψ' of additively optimal functions, there is a fixed constant $c_{\psi,\psi'}$, depending only on ψ and ψ', such that for all x, y we have

$$|C_\psi(x|y) - C_{\psi'}(x|y)| \leq c_{\psi,\psi'}.$$

To see this, first substitute $\phi_0 = \psi$ and $\phi = \psi'$ in Theorem 2.1.1, then substitute $\phi = \psi$ and $\phi_0 = \psi'$ in Theorem 2.1.1, and combine the two resulting inequalities. While the complexities according to ψ and ψ' are not exactly equal, they are *equal up to a fixed constant* for all x and y.

Definition 2.1.2 Fix an additively optimal universal ϕ_0 and dispense with the subscript by defining the *conditional Kolmogorov complexity* $C(\cdot|\cdot)$ by

$$C(x|y) = C_{\phi_0}(x|y).$$

This particular ϕ_0 is called the *reference function* for C. We also fix a particular Turing machine U that computes ϕ_0 and call U the *reference machine*. The *unconditional Kolmogorov complexity* $C(\cdot)$ is defined by

$$C(x) = C(x|\epsilon).$$

Example 2.1.1 Programmers are generally aware that programs for symbolic manipulation tend to be shorter when they are expressed in the LISP programming language than if they are expressed in FORTRAN, while for numerical calculations the opposite is the case. Or is it? The invariance theorem in fact shows that to express an algorithm succinctly in a program, it does not matter which programming language we use (up to a fixed additive constant that depends only on the two programming languages).

To see this, as an example consider the lexicographic enumeration of all syntactically correct LISP programs $\lambda_1, \lambda_2, \ldots$ and the lexicographic enumeration of all syntactically correct FORTRAN programs $\pi_1, \pi_2, \ldots$. With proper definitions we can view the programs in both enumerations as computing partial recursive functions from their inputs to their outputs. Choosing reference machines in both enumerations, we can define complexities $C_{\mathrm{LISP}}(x)$ and $C_{\mathrm{FORTRAN}}(x)$ completely analogous to $C(x)$. All of these measures of the descriptional complexity of x coincide up to a fixed additive constant. Let us show this directly for $C_{\mathrm{LISP}}(x)$ and $C_{\mathrm{FORTRAN}}(x)$.

It is well known and also easy to see that each enumeration contains a universal program; the LISP enumeration contains a LISP interpreter program that interprets any LISP program. But there is also a LISP program λ_P that is a FORTRAN interpreter in the sense that it interprets any FORTRAN program. Consequently, $C_{\mathrm{LISP}}(x) \leq C_{\mathrm{FORTRAN}}(x) + l(\lambda_P)$. Similarly, there is a FORTRAN program π_L that is a LISP interpreter, which yields $C_{\mathrm{FORTRAN}}(x) \leq C_{\mathrm{LISP}}(x) + l(\pi_L)$. Consequently, $|C_{\mathrm{LISP}}(x) - C_{\mathrm{FORTRAN}}(x)| \leq l(\lambda_P) + l(\pi_L)$ for all x. ◇

Example 2.1.2 In Theorem 2.1.1 we used a special type of universal partial recursive function, called 'additively optimal.' There are other universal partial recursive functions that are not additively optimal and for which the theorem does not hold. For example, let ϕ be the function computed by a universal Turing machine U_ϕ such that U_ϕ started on input $\langle y, \langle n, pp\rangle\rangle$ simulates T_n on input $\langle y, p\rangle$, and ϕ is not defined for inputs that are not of the form $\langle y, \langle n, pp\rangle\rangle$. (That is, if T_n computes the partial recursive function ϕ_n, then $\phi(\langle y, \langle n, pp\rangle\rangle) = \phi_n(\langle y, p\rangle)$.) Then, for all x, y, n, we have $C_\phi(x|y) \geq 2C_{\phi_n}(x|y)$. ◇

2.1.1 Two-Part Codes

It is a deep and useful fact that the shortest effective description of an object x can be expressed in terms of a *two-part code*, the first part describing an appropriate Turing machine and the second part describing the program that interpreted by the Turing machine reconstructs x. The essence of the invariance theorem is as follows: For the fixed reference universal Turing machine U, the length of the shortest program to compute x is $\min\{l(p) : U(p) = x\}$. Looking back at the proof of Lemma 2.1.1, we notice that $U(0p) = U(p)$. From the definitions it therefore follows that

$$C(x) = \min\{l(T) + l(p) : T(p) = x\} + O(1),$$

where $l(T)$ is the length of a self-delimiting encoding for a Turing machine T. This provides an alternative definition of Kolmogorov complexity (similarly, for conditional Kolmogorov complexity). The above expression for Kolmogorov complexity can be rewritten as

$$C(x) = \min\{l(T) + C(x|T) : T \in \{T_0, T_1, \ldots\}\} + O(1), \tag{2.1}$$

which emphasizes the two-part-code nature of Kolmogorov complexity, using the regular aspects of x to maximally compress. In the example

$$x = 10101010101010101010101010$$

we can encode x by a small Turing machine that computes x from the program 13. Intuitively, the Turing machine part of the code squeezes out the *regularities* in x. What is left are irregularities, or *random aspects*, of x relative to that Turing machine. The minimal-length two-part code squeezes out regularity only insofar as the reduction in the length of the description of random aspects is greater than the increase in the regularity description.

The right model is a Turing machine T among those that reach the minimum description length

$$\min_T \{l(T) + C(x|T) : T \in \{T_0, T_1, \ldots\}\}.$$

This T embodies the amount of useful information contained in x. The main remaining question is which such T to select among those that satisfy the requirement. The problem is how to separate a shortest program x^* for x into parts $x^* = pq$ such that p represents an appropriate T. This idea has spawned the 'minimum description length' principle in statistics and inductive reasoning, Section 5.4; Kolmogorov's structure functions and algorithmic (minimal) sufficient statistic, Section 5.5; and the notion of algorithmic entropy in Section 8.6.

2.1.2 Upper Bounds

Theorem 2.1.1 has a wider importance than just showing that the hierarchy of C_ϕ complexity measures contains an additively optimal one. It is also our principal tool in finding upper bounds on $C(x)$. Such upper bounds depend on the choice of reference function, and hence are proved only to within an additive constant.

Intuitively, the Kolmogorov complexity of a binary string cannot exceed its own length, because the string is obviously a (literal) description of itself.

Theorem 2.1.2 *There is a constant c such that for all x and y,*

$$C(x) \leq l(x) + c \text{ and } C(x|y) \leq C(x) + c.$$

Proof. The first inequality is supremely obvious: define a Turing machine T that copies the input to the output. Then for all x, we have $C_T(x) = l(x)$. By Theorem 2.1.1 the result follows.

To prove the second inequality, construct a Turing machine T that for all y, z computes output x on input $\langle z, y\rangle$ iff the universal reference machine U computes output x for input $\langle z, \epsilon\rangle$. Then $C_T(x|y) = C(x)$. By Theorem 2.1.1, there is a constant c such that $C(x|y) \leq C_T(x|y)+c = C(x) + c$. □

Note that the additive constants in these inequalities are fudge terms related to the reference machine U. For example, we need to indicate to the reference machine that a given description is the object itself, and this information adds a number of bits to the literal description. In Section 3.2 we will calculate the constants explicitly as 8 and 2, respectively. Let us look at some more examples in order to develop our intuition about the notion of complexity of description.

Example 2.1.3 For each finite binary string x we have $C(xx) \leq C(x) + O(1)$. Construct a Turing machine V such that $V(p) = U(p)U(p)$, for all programs p, where U is the reference machine in the proof of Theorem 2.1.1. In particular, if $U(p) = x$, then $V(p) = xx$. Let $V = T_m$ in the standard enumeration of Turing machines $T_1, T_2, \ldots$. With $\overline{m}$ denoting the self-delimiting description $1^{l(m)}0m$ of m, we have $U(\overline{m}p) = T_m(p) = xx$ and $l(\overline{m}p) = l(p) + 2l(m) + 1$. Hence, $C(xx) \leq C(x) + 2l(m) + 1$. From now on we leave the more obvious details of this type of argument for the reader to fill in. ◇

Example 2.1.4 Recall that x^R denotes the reverse of x. Clearly, the complexities of x and x^R can differ by at most a fixed constant c independent of x. That is, $|C(x) - C(x^R)| < c$ holds for all x. We can generalize this example as follows: For every total recursive function ϕ that is one-to-one there is (another) constant c such that $|C(\phi(x)) - C(x)| < c$ for all x.

In fact, if ϕ is computed by Turing machine T_n and $U(p) = x$, then there is a Turing machine V such that $V(\bar{n}p) = \phi(x)$. If $V = T_m$, then $U(\overline{m}\bar{n}p) = \phi(x)$, and therefore $|C(\phi(x)) - C(x)| < 2l(m) + 2l(n) + 2$. Similar relations hold for the conditional complexity $C(x|y)$. ◇

Example 2.1.5 Can the complexity of a pair of strings exceed the sum of the complexities of the individual strings? In other words, is C *subadditive*? Let $\langle \cdot \rangle : \mathcal{N} \times \mathcal{N} \rightarrow \mathcal{N}$ be the standard recursive bijection over the natural numbers that encodes x and y as $\langle x, y \rangle$. Define $C(x, y) = C(\langle x, y \rangle)$. That is, up to a fixed constant, $C(x, y)$ is the length of the shortest program such that U computes both x and y and a way to tell them apart. It is seductive to conjecture $C(x, y) \leq C(x) + C(y) + O(1)$, 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. One way to do this is by using input $\overline{l(p)}pq$ or input $\overline{l(q)}qp$. In this way, we can show that for all x, y, we have

$$C(x, y) \leq C(x) + C(y) + 2 \log(\min(C(x), C(y))). \tag{2.2}$$

We cannot eliminate the logarithmic error term for the general case. Namely, in Example 2.2.3 on page 118 we show that there is a constant

c such that for all n there are x and y of length at most n such that

$$C(x,y) \geq C(x) + C(y) + \log n - c.$$

We can eliminate the logarithmic error term at the cost of entering the length of one of the programs in the conditional,

$$C(x,y|C(x)) \leq C(x) + C(y) + O(1).$$

Equation 2.2 also holds if we replace the left-hand side by the complexity $C(xy)$ of the unmarked concatenation xy. In the example already referred to above, it is shown that we cannot eliminate the logarithmic error in this case either. $\diamond$

Example 2.1.6 If we know $C(x)$ and x, then we can run all programs of length $C(x)$ in parallel on the reference machine U in dovetail fashion (in stage k of the overall computation execute the ith computation step of program $k-i$). By definition of $C(\cdot)$, there must be a program of length $C(x)$ that halts with output x. The first such program is the *first shortest program* for x in enumeration order, and is denoted by x^*.

Therefore, a program to compute $C(x)$, given x, can be converted to a program to compute x^*, given x, at the cost of a constant number of extra bits. If we have computed x^*, then $C(x)$ is simply its length, so the converse is trivial. Furthermore, to describe $C(x)$ from scratch takes at least as many bits as to describe $C(x)$ using x. Altogether we have, up to additional constant terms,

$$C(x^*|x) = C(C(x)|x) \leq C(C(x)) \leq \log l(x).$$

$\diamond$

The upper bound on $C(x^*|x)$ cannot be improved to $O(1)$. If it could, then one could show that $C(x)$ is a recursive function. However, in Theorem 2.3.2 we shall show that $C(x)$ is not partial recursive. It is a curious fact that for some x, knowledge of x does not help much in computing x^*. In fact, the upper bound is nearly optimal. In Theorem 3.8.1 we shall show that for some x of each length n the quantity $C(C(x)|x)$, and hence also $C(x^*|x)$, is almost $\log n$.

Clearly, the information that an element belongs to a particular set can severely curtail the complexity of that element. The following simple observation, due to Kolmogorov, turns out to be very useful. We show that for every easily describable set the conditional complexity of every one of its elements is at most equal to the logarithm of the cardinality of that set. (We will observe later, in Theorem 2.2.1, that the conditional complexities of the majority of elements in a finite set cannot be significantly less than the logarithm of the cardinality of that set: we will say that they are ‘incompressible’ and have a small ‘randomness deficiency.’)

Theorem 2.1.3 *Let $A \subseteq \mathcal{N} \times \mathcal{N}$ be recursively enumerable, and $y \in \mathcal{N}$. Suppose $Y = \{x : (x, y) \in A\}$ is finite. Then, for some constant c depending only on A, for all x in Y, we have $C(x|y) \leq l(d(Y)) + c$.*

Proof. Let A be enumerated without repetition as $(x_1, y_1), (x_2, y_2), \ldots$ by a Turing machine T. Let $(x_{i_1}, y_{i_1}), \ldots, (x_{i_k}, y_{i_k})$ be the subsequence in which the elements of Y are enumerated, $k = d(Y)$. Using the fixed y, modify T to T_y such that T_y, on input $1 \leq p \leq d(Y)$, outputs x_{i_p}, $T_y(p) = x_{i_p}$. Therefore, we have by the invariance theorem, Theorem 2.1.1, that $C(x|y) \leq C_{T_y}(x) + c \leq l(d(Y)) + c$, with c depending only on A. □

Let us illustrate the use of this theorem. Let A be a subset of $\mathcal{N}$. Define $A^{\leq n} = \{x \in A : l(x) \leq n\}$. Let A be recursively enumerable and $d(A^{\leq n}) \leq p(n)$, with p a polynomial. Then, for all $x \in A$ of length at most n we have $C(x|n) \leq l(p(n)) + O(1)$, by Theorem 2.1.3. For all x of length at most n we have $C(x) \leq C(x|n) + 2l(n) + O(1)$. Therefore, for $x \in A^{\leq n}$ we find that $C(x) = O(\log n)$.

2.1.3 Invariance of Kolmogorov Complexity

The complexity $C(x)$ is invariant only up to a constant depending on the reference function ϕ_0. Thus, one may object, for *every* string x there is an additively optimal recursive function ψ_0 such that $C_{\psi_0}(x) = 0$. So how can one claim that $C(x)$ is an objective notion?

A mathematically clean solution to this problem is as follows: Call two complexities C_ϕ and C_ψ *equivalent*, $C_\phi \equiv C_\psi$, if there is a constant c such that for all x,

$$|C_\phi(x) - C_\psi(x)| \leq c.$$

Then the equivalence relation $\equiv$ induces equivalence classes

$$[C_\phi] = \{C_\psi : C_\psi \equiv C_\phi\}.$$

We order the equivalence classes by $[C_\phi] \leq [C_\psi]$ if there is a constant $c \geq 0$ such that $C_\phi(x) \leq C_\psi(x) + c$ for every x. The resulting order on the equivalence classes is a partial order with a single minimal element, namely $[C_{\phi_0}]$, such that for all C_ψ,

$$[C_{\phi_0}] \leq [C_\psi].$$

We have somewhat glibly overlooked the fact that our definition of Kolmogorov complexity is relative to the particular effective enumeration of Turing machines as used in the proof of the invariance theorem, Theorem 2.1.1. We have claimed that the quantity of information in an object depends on itself alone. That is, it should be independent of the particular enumeration of Turing machines.

Consider two different enumerations of all partial recursive functions, say $\phi_1, \phi_2, \ldots$ and $\psi_1, \psi_2, \ldots$. Assume that the ϕ enumeration is the enumeration corresponding to our effective enumeration of Turing machines as used in the proof of the invariance theorem.

Let the standard enumeration $\phi_1, \phi_2, \ldots$ and the other enumeration $\psi_1, \psi_2, \ldots$ be related by $\psi_i = \phi_{f(i)}$ and $\phi_i = \psi_{g(i)}$, $i = 1, 2, \ldots$. If both f and g are partial recursive, then the enumerations are called *recursively isomorphic* and are both *acceptable numberings* (Section 1.7, Exercise 1.7.6 on page 41). Let $C(x)$ be the complexity with respect to the reference function in the ϕ enumeration, and let $C'(x)$ be the complexity with respect to the reference function in the ψ enumeration. It is an easy exercise to show that there is a constant c such that $|C(x) - C'(x)| < c$ for all x. (Hint: use the indexes of f and g in the enumerations.)

Therefore, not only do additively optimal functions in the *same* acceptable numberings yield complexities that are equal up to a fixed constant, but additively optimal functions in two *different* acceptable numberings do so as well. Hence, Kolmogorov complexity is *recursively invariant* between acceptable numberings, even though we have chosen to define it using the specific enumeration of Turing machines of Section 1.7. Using an analogy due to Hartley Rogers, Jr., the fixed choice of effective enumeration of Turing machines can be compared with using a particular coordinate system to establish coordinate-free results in geometry.

A contradiction is possible only if there is no recursive isomorphism between the ϕ enumeration and the ψ enumeration. We give an example of an enumeration of all partial recursive functions for which an additively optimal function yields a complexity $C'(x)$ such that $|C(x) - C'(x)|$ is unbounded. Let $C(x)$ be defined with respect to the ϕ enumeration as in Theorem 2.1.1. Define the ψ enumeration as follows: The even functions ψ_{2i} are defined by $\psi_{2i}(1) := y_i$ for some y_i with $C(y) \geq i^2$ and $\psi_{2i}(x) := \phi_i(x)$ for all $x > 1$. The odd functions ψ_{2i+1} are given by $\psi_{2i+1} := \phi_i$.

Clearly, the ψ enumeration contains all partial recursive functions. By way of contradiction, assume that $C'(\cdot)$ is the Kolmogorov complexity in the ψ-enumeration defined as in Theorem 2.1.1. Then, $C'(y_i) \leq C'_{\psi_{2i}}(y_i) + c_{\psi_{2i}}$. By construction, $C'_{\psi_{2i}}(y_i) = 1$ and $c_{\psi_{2i}} \leq 2 \log 2i + O(1)$. On the other hand, $C(y_i) > i^2$ by construction. Hence, $|C'(y_i) - C(y_i)|$ rises unboundedly with i.

2.1.4 Concrete Kolmogorov Complexity

It is possible to eliminate the indeterminacy of 'equality up to a constant' everywhere by using a fixed domain of objects, a fixed effective enumeration of Turing machines, and a fixed choice of additively optimal function (rather the universal Turing machine that computes it). Start from the enumeration of Turing machines in Section 1.7. Fix any small universal machine, say U, with state–symbol product less than 30. There exists at least one 7×4 universal Turing machine as mentioned in the comment on page 31 following Example 1.7.4. In Section 3.2 we exhibit a universal reference machine U to fix a concrete Kolmogorov complexity with $C(x|y) \leq l(x) + 2$ and $C(x) \leq l(x) + 8$.

For every x it is of course possible to choose a universal Turing machine U' such that $C_{U'}(x) = 0$ (in this notation identifying U' with the function it computes). For every such universal Turing machine U', we have for all x that

$$C(x) \leq C_{U'}(x) + C(U').$$

Here $C(U')$ is at least the length of the shortest program p such that for all programs q we have $U(pq) = U'(q)$. This means that if $C_{U'}(x) = 0$, then $C(U') \geq C(x)$. That is, $C_{U'}(x) = 0$ unavoidably means that the description of U' contains a description of x. Therefore, in order to assign low complexity to a large and complicated object, a universal machine has to be large and complicated as well.

Exercises

2.1.1. [15] (a) Show that $C(0^n|n) \leq c$, where c is a constant independent of n.

(b) Show that $C(\pi_{1:n}|n) \leq c$, where $\pi = 3.1415\ldots$ and c is some constant independent of n.

(c) Show that we can expect $C(a_{1:n}|n) \leq \frac{1}{4}n$, where a_i is the ith bit in Shakespeare's *Romeo and Juliet.*

(d) What is $C(a_{1:n}|n)$, where a_i is the ith bit in the expansion of the fine structure constant $a = e^2/\hbar c$, in physics.

Comments. Hint: for Item (c) use known facts concerning the letter frequencies (entropy) in written English. Source: T.M. Cover, *The Impact of Processing Technique on Communications,* J.K. Skwirzynski, ed., Martinus Nijhof, 1985, pp. 23–33.

2.1.2. [10] Let x be a finite binary string with $C(x) = q$. What is the complexity $C(x^q)$, where x^q denotes the concatenation of q copies of x?

2.1.3. [14] Show that there are infinite binary sequences ω such that the length of the shortest program for reference Turing machine U to compute the consecutive digits of ω one after another can be significantly shorter than the length of the shortest program to compute an initial n-length segment $\omega_{1:n}$ of ω, for any large enough n.

Comments. Hint: choose ω a recursive sequence with shortest program of length $O(1)$. Then $C(\omega_{1:n}) = C(n) + O(1)$, which goes to ∞ with n.

2.1.4. [12] Prove that for every x, there is an additively optimal function ϕ_0 (as in Theorem 2.1.1) such that $C_{\phi_0}(x) = 0$. Prove the analogous statement for x under condition y.

2.1.5. [07] Below, x, y, and z are arbitrary elements of $\mathcal{N}$. Prove the following:

(a) $C(x|y) \leq C(x) + O(1)$.

(b) $C(x|y) \leq C(x, z|y) + O(1)$.

(c) $C(x|y, z) \leq C(x|y) + O(1)$.

(d) $C(x, x) = C(x) + O(1)$.

(e) $C(x, y|z) = C(y, x|z) + O(1)$.

(f) $C(x|y, z) = C(x|z, y) + O(1)$.

(g) $C(x, y|x, z) = C(y|x, z) + O(1)$.

(h) $C(x|x, z) = C(x|x) + O(1) = O(1)$.

2.1.6. [14] Let ϕ_k be any partial recursive function in the effective enumeration $\phi_1, \phi_2, \ldots$. Let x, y, z be arbitrary elements of $\mathcal{N}$. Prove the following:

(a) $C(\phi_k(x)|y) \leq C(x|y) + 2l(k) + O(1)$.

(b) $C(y|\phi_k(x)) \geq C(y|x) - 2l(k) + O(1)$.

Assume that ϕ_k is also one-to-one. Show that

(c) $|C(x) - C(\phi_k(x))| \leq 2l(k) + O(1)$.

(d) $C(x|y, z) \leq C(x|\phi_k(y), z) + 2l(k) + O(1)$.

2.1.7. [12] Let x, y, z, and ϕ_k be as before. Prove the following.

(a) $C(x, y) \leq C(x) + 2l(C(x)) + C(y|x) + O(1)$.

(b) $C(\phi_k(x, y)) \leq C(x) + 2l(C(x)) + C(y|x) + 2l(k) + O(1) \leq C(x) + 2l(C(x)) + C(y) + 2l(k) + O(1)$.

2.1.8. [12] Show that if ϕ is a fixed one-to-one and onto recursive function $\phi : \{0, 1\}^* \rightarrow \{0, 1\}^*$, then for every $x \in \{0, 1\}^*$,

$$C(x) - C(x|\phi(x)) = C(x) + O(1) = C(\phi(x)) + O(1).$$

2.1.9. • [19] We investigate the invariance of C under change of program representations from 2-ary to r-ary representations. Let $A_r = \{0, 1, \ldots, r-1\}^*$, $r \geq 2$, and $A = \mathcal{N}^*$ with $\mathcal{N}$ the set of natural numbers. A function $\phi : A_r \times A \rightarrow A$ is called an r-ary *decoder*. In order not to hide too much information in the decoder, we want it to be a simple function, a partial recursive one. Analogous to the definitions in the main text, for any binary decoder ϕ and x, y in A,

$$C_\phi(x|y) = \min\{l(p) : \phi(p, y) = x\},$$

or ∞ if such p does not exist.

(a) Prove Theorem 2.1.1 under this definition of C.

(b) Define for each pair of natural numbers $r, s \geq 2$ a standard encoding E of strings x in base r to strings $E(x)$ in base s such that $l(E(x)) \leq l(x) \log r / \log s + 1$.

(c) Prove the invariance theorem, Theorem 2.1.1, for r-ary decoders ϕ. First, let us define $C_\phi(x|y) = \min\{l(p) \log r : \phi(p, y) = x\}$ and $C_\phi(x|y) = \infty$ if such p does not exist. Then prove that there exists an additively optimal (universal) r-ary decoder ϕ_0 such that for all s, for all s-ary decoders ϕ, there exists a constant c_ϕ such that for all $x, y \in A$ we have

$$C_{\phi_0}(x|y) \leq C_\phi(x|y) + c_\phi.$$

(d) Show that for any $x \in A_r$ of length n, we have $C(x) \leq n \log r + 2 \log r + c$ for some fixed c, independent of x and r.

(e) Fix natural numbers $r, s \geq 2$ and choose an additively optimal r-ary decoder and an additively optimal s-ary decoder. Call the associated canonical C measures respectively C_r and C_s. Show that there exists a constant c such that for all x, y in A we have

$$|C_r(x|y) - C_s(x|y)| \leq c,$$

where c is independent of x and y. Conclude that C_2, the C measure treated in the main text, is universal in the sense that neither the restriction to binary objects to be described nor the restriction to binary descriptions (programs) results in any loss of generality.

Comments. In general, if we denote by $C_r(x)$ the analogous complexity of x in terms of programs over alphabets of r letters ($C_2(x) = C(x)$ but for $r > 2$ without the $\log r$ normalizing multiplicative factor as in Item (c)), then by the same analysis as of Item (c) we obtain $C_r(x) \sim C(x)/\log r$. Source: P. Gács, *Lecture Notes on Descriptional Complexity and Randomness*, Manuscript, Boston University, 1987.

2.1.10. [12] (a) Show that $C(x + C(x)) \leq C(x) + O(1)$.

(b) Show that if $m \leq n$, then $m + C(m) \leq n + C(n) + O(1)$.

Comments. Hint for Item (a): if $U(p) = x$ with $l(p) = C(x)$, then p also suffices to reconstruct $x + l(p)$. Hint for Item (b): use Item (a). Source: P. Gács, *Lecture Notes on Descriptional Complexity and Randomness*, Manuscript, Boston University, 1987; result is attributed to C.P. Schnorr.

2.1.11. [13] Let $\phi_1, \phi_2, \ldots$ be the standard enumeration of the partial recursive functions, and let a be a fixed natural number such that the set $A = \{x : \phi_k(y) = \langle a, x \rangle$ for some $y \in \mathcal{N}\}$ is finite. Show that for each x in A we have $C(x|a) \leq l(d(A)) + 2l(k) + O(1)$.

2.1.12. [18] Define the *function complexity* of a function $f : \mathcal{N} \rightarrow \mathcal{N}$, restricted to a finite domain D, as

$$C(f|D) = \min\{l(p) : \forall_{x \in D}[U(p,x) = f(x)]\}.$$

(a) Show that for all recursive functions f, there exists a constant c_f such that for all finite $D \subseteq \mathcal{N}$, we have $C(f|D) \leq c_f$.

(b) Show that for all partial recursive functions, for all $D = \{i : i \leq n\}$, we have $C(f|D) \leq \log n + c_f$, where c_f depends on f but not on D.

Comments. Compare Theorem 2.7.2. Source: J.M. Barzdins, *Soviet Math. Dokl.*, 9(1968), 1251–1254.

2.2 Incompressibility

It is easy to see that there are strings that can be described by programs much shorter than themselves. For instance, the function defined by $f(1) = 2$ and $f(i) = 2^{f(i-1)}$ for $i > 1$ grows very fast, $f(k)$ is a stack of k twos. Yet for each k it is clear that the string $x = 1^{f(k)}$, or the integer $y = 2^{f(k)}$, has at most complexity $C(k) + c$ for some constant c independent of k.

Trivially, this simple argument can be generalized to prove the following fact: for every recursive function ϕ, no matter how fast it grows, there is a constant c such that for each value of n there is a string x such that $l(x) = \phi(n)$ but $C(x) \leq n + c$. That is, for an appropriate sequence of strings, the ratio of string length to description length can increase as fast as any recursive function—some strings are very *compressible.*

What about incompressibility? By a simple counting argument one can show that whereas some strings can be greatly compressed, the majority of strings cannot be compressed at all.

For each n there are 2^n binary strings of length n, but only $\sum_{i=0}^{n-1} 2^i = 2^n - 1$ possible shorter descriptions. Therefore, there is at least one binary string x of length n such that $C(x) \geq n$. 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 $C(x|y) \geq n$.

Definition 2.2.1 For each constant c we say that a string x is *c-incompressible* if $C(x) \geq l(x) - c$.

Strings that are incompressible (say, c-incompressible with small c) are patternless, since a pattern could be used to reduce the description length. Intuitively, we think of such patternless sequences as being random, and we use ‘random sequence’ synonymously with ‘incompressible sequence.’ Later we give a formalization of the intuitive notion of a random sequence as a sequence that passes all effective tests for randomness.

How many strings of length n are c-incompressible? By the same counting argument we find that the number of strings of length n that are c-incompressible is at least $2^n - 2^{n-c} + 1$. Hence there is at least one 0-incompressible string of length n, at least one-half of all strings of length n are 1-incompressible, at least three-fourths of all strings of length n are 2-incompressible, ..., and at least the $(1 - 1/2^c)$th part of all 2^n strings of length n are c-incompressible. This means that for each constant $c > 1$ the majority of all strings of length n with $n > c$ are c-incompressible. We generalize this to the following simple but extremely useful *incompressibility theorem.*

Theorem 2.2.1 *Let c be a positive integer. For each fixed y, every finite set A of cardinality m has at least $m(1-2^{-c})+1$ elements x with $C(x|y) \geq \log m - c$.*

Proof. The number of programs of length less than $\log m - c$ is

$$\sum_{i=0}^{\log m-c-1} 2^i = 2^{\log m-c} - 1.$$

Hence, there are at least $m - m2^{-c} + 1$ elements in A that have no program of length less than $\log m - c$. □

As an example, set $A = \{x : l(x) = n\}$. Then the cardinality of A is $m = 2^n$. Since Theorem 2.1.2 asserts that $C(x) \leq n + c$ for some fixed c and all x in A, Theorem 2.2.1 demonstrates that this trivial estimate is quite sharp. The deeper reason is that since there are few short programs, there can be only few objects of low complexity.

It is important to realize that Theorem 2.1.1 and Theorem 2.2.1, together with the trivial upper bound of Theorem 2.1.2, give us already all we need for most applications.

Example 2.2.1 Are all substrings of incompressible strings also incompressible? A string $x = uvw$ of length n can be specified by a short program p for v and the string uw itself. Additionally, we need information on how to tell these items apart. For instance, $q = \overline{l(p)}p\overline{l(u)}uw$ is a program for x. There exists a machine T that starting on the left end of q, first determines $l(p)$, then uses $l(p)$ to delimit p, and computes v from p. Continuing on its input, T determines $l(u)$ and uses this to delimit u on the remainder of its input. Subsequently, T reassembles x from the three pieces v, u, and w it has determined. It follows that $C(x) \leq C_T(x) + O(1) \leq l(q) + O(1)$, since $l(q) \leq C(v) + 2l(C(v)) + 2l(n) + n - l(v) + 2$. Therefore,

$$C(x) \leq C(v) + n - l(v) + 4\log n + O(1).$$

Hence, for c-incompressible strings x with $C(x) \geq n - c$ we obtain

$$C(v) \geq l(v) - O(\log n).$$

Thus, we have shown that v is incompressible up to an additive term logarithmic in n.

Can we hope to prove $C(v) \geq l(v) - O(1)$ for all x and v? If this were true, then x could not contain long regular subsequences, for instance, a subsequence of k zeros has complexity $O(\log k)$ and not $k - O(1)$. However, the very restriction on x of not having long regular subsequences imposes some regularity on x by making it a member of a relatively small set. Namely, we can describe x by stating that it does not contain certain subsequences, followed by x's index in the set that is determined by these constraints. But the set of which x is a member is so small that $C(x)$ drops below $n - c$, and x is compressible. Hence, the very incompressibility of x requires that it have compressible substrings. This corresponds to a fact we know from probability theory: a random sequence must contain long runs of zeros. ◇

Example 2.2.2 If p is a shortest program for x, so that $C(x) = l(p)$, then we would like to assert that p is incompressible. This time, our intuition corresponds with the truth. There is a constant $c > 0$ such that for all strings x we have $C(p) \geq l(p) - c$. For suppose the contrary, and for every constant c there is an x and a shortest program q that generates p with $l(q) < l(p) - c$. Define a universal machine V that works just like the reference machine U, except that V first simulates U on its input to obtain the output, and then uses this output as input on which to simulate U once more. Let $V = T_i$, the ith Turing machine in the standard enumeration. Then, U with input $1^i 0q$ computes x, and therefore $C(x) < l(p) - c + i + 1$. But this contradicts $l(p) = C(x)$ for $c \geq i + 1$. ◇

Example 2.2.3 We continue Example 2.1.5 on page 109 that $C(x, y)$ is not *subadditive* since the logarithmic term in Equation 2.2 cannot be eliminated. Namely, there are $(n + 1)2^n$ pairs (x, y) of binary strings whose sum of lengths is n. By Theorem 2.2.1 there is a pair (x, y) with $l(x) + l(y) = n$ such that $C(x, y) \geq n + \log n - 1$. But Theorem 2.1.2 on page 108 states that $C(x) + C(y) \leq l(x) + l(y) + c$ for some constant c independent of x and y. Hence, for all n there are x and y of length at most n such that

$$C(x, y) > C(x) + C(y) + \log n - c,$$

where c is a constant independent of x and y. For the unmarked concatenation xy with $l(xy) = n$, if $C(xy) \geq n$, then xy contains a block of 0's or 1's of length at least $\log n - 2 \log \log n - O(1)$ (follows from Example 2.2.1 but is more precisely derived in Corollary 2.6.2 om page 172). We can choose the concatenation xy so that x ends with this longest run of 0's or 1's. This means that $C(x) \leq l(x) - \log n + 2 \log \log n$. Then, $C(xy) \geq C(x) + C(y) + \log n - 2 \log \log n - c$. ◇

There is a particular use we had in mind in defining conditional Kolmogorov complexity. Namely, we often want to speak about the complexity of x given its length n. This is because a string x of length n carries in a sense *two* quantities of information, one associated with the irregularity of the pattern of 0's and 1's in x, and one associated with the length n of x.

Example 2.2.4 One effect of the information quantity associated with the length of strings is that $C(x)$ is *nonmonotonic on prefixes*. This can be due to the information contained in the length of x. That is, for $m < n$ we can still have $C(m) > C(n)$. But then $C(y) > C(x)$ for $x = 1^n$ and $y = 1^m$, notwithstanding that y is a proper prefix of x. For example, if $n = 2^k$, then $C(1^n) \leq \log\log n + O(1)$, while Theorem 2.2.1 shows that there exist m with $\frac{1}{2}n \leq m < n$ such that $C(1^m) \geq \log n - O(1)$. Therefore, the complexity of a part can turn out to be bigger 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. ◇

Definition 2.2.2 The *length-conditional* Kolmogorov complexity of x is $C(x|l(x))$.

Roughly speaking, this means that the length of the shortest program for x may save up to $\log l(x)$ bits in comparison with the shortest program in the unconditional case. Clearly, there is a constant c such that for all x,

$$C(x|l(x)) \leq C(x) + c.$$

While on the face of it the measure $C(x|l(x))$ gives a pure estimate of the quantity of information in solely the pattern of 0's and 1's of x, this is not always true. Namely, sometimes the information contained in $l(x)$ can be used to determine the pattern of zeros and ones of x. This effect is noticeable especially in the low-complexity region.

Example 2.2.5 For each integer n, the *n-string* is defined by $n0^{n-l(n)}$ (using the binary string n). There is a constant c such that for all n, if x is the n-string, then $C(x|n) \leq c$. Namely, given n we can find the nth binary string according to Equation 1.3 and pad the string with zeros up to overall length n. We use n-strings to show that unfortunately, like the original $C(x)$, the complexity measure $C(x|l(x))$ is *not monotonic over the prefixes.* Namely, if we choose n such that its pattern of 0's and 1's is very irregular, $C(n) \geq l(n)$, then for $x = n0^{n-l(n)}$, we still obtain $C(x|l(x)) \leq c$. But clearly $C(n|l(n)) \geq C(n) - C(l(n)) \geq \log n - 2\log\log n$. ◇

Example 2.2.6 Consider the complexity of a string x, with x an element of a given set A. Clearly, the information that an element belongs to a particular set severely curtails the complexity of that element if that set is small or sparse. The following is a simple application of the very useful Theorem 2.1.3. Let A be a subset of $\mathcal{N}$ and $A^{\leq n} = \{x \in A : l(x) \leq n\}$ We call A *meager* if $\lim d(A^{\leq n})/2^n = 0$ for $n \to \infty$. For example, the set of all finite strings that have twice as many 0's as 1's is meager. We show that meagerness may imply that almost all strings in the meager set have short descriptions.

Claim 2.2.1 If A is recursive and meager, then for each constant c there are only finitely many x in A that are c-incompressible ($C(x) \geq l(x) - c$).

Proof. Consider the lexicographic enumeration of all elements of A. Because A is recursive, there is a total recursive function ϕ_i that enumerates A in increasing order. Hence, for the jth element x of A we have $C(x) \leq C(j) + 2l(i) + 1$. If x has length n, then the meagerness of A implies that for each constant c', no matter how large, $n - C(j) > c'$ from some n onward. Hence, $C(x) < n - c' + 2l(i)$. The proof is completed by setting $c' = c + 2l(i)$. □ ◇

2.2.1 Randomness Deficiency

If we know that x belongs to a subset A of the natural numbers, then we can consider its complexity $C(x|A)$. For instance, $C(x) = C(x|\mathcal{N})$, because it is understood that x is a natural number. If x is an element of a finite set A, then Theorem 2.1.3 asserts that $C(x|A) \leq l(d(A)) + c$ for some c independent of x but possibly dependent on A. For instance, the infinite meager sets of Example 2.2.6 contain finitely many incompressible strings only.

Definition 2.2.3 The *randomness deficiency* of x *relative* to A is defined as $\delta(x|A) = l(d(A)) - C(x|A)$. It follows that $\delta(x|A) \geq -c$ for some fixed constant c independent of x.

If $\delta(x|A)$ is large, then this means that there is a description of x with the help of A that is considerably shorter than just giving x's serial number in A. There are comparatively few objects with large randomness deficiency—this is the substance of Martin-Löf's notion of a statistical test in Section 2.4. Quantitatively this is expressed as follows:

Theorem 2.2.2 *Assume the discussion above. Then,* $d(\{x : \delta(x|A) \geq k\}) \leq d(A)/2^{k-1}$.

Proof. There are fewer than 2^{l+1} descriptions of length at most l. □

By Theorem 2.1.3, the complexity of a string x in a given finite section of a recursively enumerable set is bounded above by the logarithm of the cardinality of that finite section. Let $\langle\cdot\rangle : \mathcal{N}^2 \to \mathcal{N}$ be the standard recursive bijective pairing function. Let $R = \{(x, y) : \phi(i) = \langle x, y\rangle, i \geq 1\}$ with ϕ a partial recursive function, say $\phi = \phi_r$ in the standard enumeration of partial recursive functions. Then R is recursively enumerable. Let the set $A = \{x : (x, y) \in R\}$ be finite. We can assume that A is enumerated without repetition, and that $j \leq d(A)$ is the position of x in this enumeration. Clearly,

$$C(x|y) \leq \log d(A) + \log r + 2 \log\log r + O(1).$$

As above, define $C(x|A) = C(x|y)$ with the obvious interpretation. The randomness deficiency of x relative to y is

$$\delta(x|y) = \log d(A) - C(x|y).$$

The randomness deficiency measures the difference between the maximal complexity of a string in A and the complexity of x in A. Now, the defect of randomness is positive up to a fixed constant independent of x and A (but dependent on r). We may consider x to be random in the set A iff $\delta(x|y) = O(1)$. If A is the set of binary strings of length n, or equivalently, R is the set $\{(x, n) : l(x) = n\}$ and $A = \{x : l(x) = n\}$, then we note that

$$\delta(x|n) = n - C(x|n) + O(1).$$

That is, x is a random finite string in our informal sense iff $\delta(x|n) = O(1)$. It will turn out that this coincides with Martin-Löf's notion of randomness in Section 2.4.

Exercises

2.2.1. [08] Prove the following continuity property of $C(x)$. For all natural numbers x, y we have $|C(x + y) - C(x)| \leq 2l(y) + O(1)$.

2.2.2. [15] Let x satisfy $C(x) \geq n - O(1)$, where $n = l(x)$.

(a) Show that $C(y), C(z) \geq \frac{1}{2}n - O(1)$ for $x = yz$ and $l(y) = l(z)$.

(b) Show that $C(y) \geq n/3 - O(1)$ and $C(z) \geq 2n/3 - O(1)$ for $x = yz$ and $l(z) = 2l(y)$.

(c) Let $x = x_1 \ldots x_{\log n}$ with $l(x_i) = n/\log n$ for all $1 \leq i \leq \log n$. Show that $C(x_i) \geq n/\log n - O(\log\log n)$ for all $1 \leq i \leq \log n$.

2.2.3. [21] Let x satisfy $C(x) \geq n - O(1)$, where $n = l(x)$. Show that for all divisions $x = yz$ we have $n - \log n - 2\log\log n \leq C(y) + C(z)$ and for some divisions we have $C(y) + C(z) \leq n - \log n + \log\log n$.

2.2.4. [23] Assume that the elements of $\{1, \ldots, n\}$ are uniformly distributed with probability $1/n$. Compute the expected value of $C(x)$ for $1 \leq x \leq n$.

Comments. Hint: $\sum_{x=1}^{n} \frac{C(x)}{n} \geq \sum_{i=1}^{\log n} 2^{-i}(1 - \frac{i}{\log n}) = \log n + O(1)$.

2.2.5. [14]We call x an *n-string* if x has length n and $x = n00\ldots0$.

(a) Show that there is a constant c such that for all n, every n-string x has complexity $C(x|n) \leq c$. (Of course, c depends on the reference Turing machine U used to define C.)

(b) Show there is a constant c such that for all n, $C(x|n) \leq c$ for every x in the form of the n-length prefix of $nn\ldots n$.

(c) Let c be as in Item (a). Consider some n and some string x of length n with $C(x|n) \gg c$. Prove that the extension of x to a string $y = x00\ldots0$ of length x has complexity $C(y|x) \leq c$. Conclude that there is a constant c such that each string x, no matter how high its $C(x|l(x))$ complexity, can be extended to a string y with $C(y|l(y)) \leq c$.

Comments. The $C(x)$ measure contains the information about the *pattern* of 0's and 1's in x and information about the *length* n of x. For random such n, the complexity $C(n) = l(n) + O(1)$ is about $\log n$. In this case, about $\log n$ bits of the shortest program p for x will be used to account for x's length. For n's that are easy to compute, this is much less. This seems a minor problem at high complexities, but becomes an issue at low complexities, as follows. If the quantities of information related to the *pattern only* is low, say less than $\log n$, for two strings x and y of length n, then distinctions between these quantities for x and y may get blurred in the comparison between $C(x)$ and $C(y)$ if the quantity of information related to length n dominates in both. The $C(x|l(x))$ complexity was meant to measure the information content of x apart from its length. However, as the present exercise shows, in that case $l(x)$ may contain already the complete description of x up to a constant number of bits. Source: D.W. Loveland, *Inform. Contr.*, 15(1969), 510–526.

2.2.6. [19] (a) Show that there is a constant $d > 0$ such that for every n there are at least $\lfloor 2^n/d \rfloor$ strings x of length n with $C(x|n) \geq n$ and $C(x) \geq n$.

(b) Show that there are constants $c, d' > 0$ such that for every large enough n there are at least $\lfloor 2^n/d' \rfloor$ strings x of length $n - c \leq l(x) \leq n$ with $C(x|n) > n$ and $C(x) > n$.

(c) Assume that we have fixed a reference universal turing machine such that for every n, we have $C(x), C(x|n) \leq n+1$ for all strings x of length n. Show that in this case Item (b) holds with $l(x) = n$.

Comments. Hint for Item (a): There is a constant $c > 0$ such that for every n and every x of length $l(x) \leq n - c$ we have $C(x|n) \leq n$ by Theorem 2.1.2. Therefore, there are at most $2^n - 2^{n-c+1}$ programs of length $< n$ available as shortest programs for the strings of length n. Hence there is at least one x of length n with $C(x|n) \geq n$. Let there be $m \geq 1$ such strings. Given m and n we can enumerate all $2^n - m$ strings x of length n and complexity $C(x|n) < n$ by dovetailing the running of all programs of length $< n$. The lexicographic first string of length n not in the list satisfies $\log m + O(1) \geq C(x|n) \geq n$. The unconditional result follows similarly by padding the description of x up to length n. Hint for Item (b): For every n there are equally many strings of length $\leq n$ to be described and potential programs of length $\leq n$ to describe them. Since some programs do not halt (Lemma 1.7.5 on page 34) for every large enough n, there exists a string x of length at most n that has $C(x|n), C(x) > n$ (and $C(x|n), C(x) \leq l(x) + c$). The remaining argument is similar to that of Item (a). Source: H. Buhrman, T. Jiang, M. Li, P.M.B. Vitányi, *Theoret. Comput. Sci.*, 235:1(2000), 59–70. Also reported by M. Kummer and L. Fortnow. Compare with the similar Exercise 3.3.1 for prefix Kolmogorov complexity on page 213. In the source of that exercise, some form of the result of the current exercise is attributed to G.J. Chaitin in the early 1970s.

2.2.7. [14] We can extend the notion of c-incompressibility as follows (all strings are binary): Let $g : \mathcal{N} \to \mathcal{N}$ be unbounded. Call a string x of length n *g-incompressible* if $C(x) \geq n - g(n)$. Let $I(n)$ be the number of strings x of length at most n that are g-incompressible. Show that $\lim_{n\to\infty} I(n)/2^{n+1} = 1$.

Comments. Thus, the g-incompressible finite strings have uniform probability going to 1 in the set of strings of length n for $n \to \infty$.

2.2.8. [19] Prove that for each binary string x of length n there is a y equal to x except for one bit such that $C(y|n) \leq n - \log n + O(1)$.

Comments. Hint: the set of binary strings of length n constituting a Hamming code has $2^n/n$ elements and is recursive. Source: personal communication, I. Csiszár, May 8, 1993.

2.2.9. [12] A Turing machine T computes an infinite sequence ω if there is a program p such that $T(p, n) = \omega_{1:n}$ for all n. Define $C(\omega) = \min\{l(p) : U(p, n) = \omega_{1:n} \text{ for all } n\}$, or ∞ if such a p does not exist. Obviously, for all ω either $C(\omega) < \infty$ or $C(\omega) = \infty$.

(a) Show that $C(\omega) < \infty$ iff $0.\omega$ is a *recursive real number* as in Exercise 1.7.22 on page 47. For the mathematical constants π and e, $C(\pi) < \infty$ and $C(e) < \infty$.

(b) Show that the reals $0.\omega$ with $C(\omega) < \infty$ form a countably infinite set and that the reals $0.\omega$ with $C(\omega) = \infty$ have uniform measure one in the total set of reals in the interval $[0, 1)$.

2.2.10. [27] We consider how information about x can be dispersed. Let $x \in \mathcal{N}$ with $l(x) = n$ and $C(x) = n + O(1)$. Show that there are $u, v, w \in \mathcal{N}$ such that

(i) $l(u) = l(v) = l(w) = \frac{1}{2}n$, $C(u) = C(v) = C(w) = \frac{1}{2}n$ $(+O(1))$, and they are pairwise independent: $C(y|z) = \frac{1}{2}n + O(1)$ for $y, z \in \{u, v, w\}$ and $y \neq z$;

(ii) x can be reconstructed from any two of them: $C(x|y, z) = O(1)$, where $y, z \in \{u, v, w\}$ and $y \neq z$.

Can you give a general solution for finding $m+k$ elements of $\mathcal{N}$ such that each of them has length and complexity n/m, and x can be reconstructed from any m distinct elements?

Comments. It is surprising that x can be reconstructed from any two out of three elements, each of one-half the complexity of x. This shows that the identity of the individual bits is not preserved in the division. Hint: assume $n = 2m$ and $x = x_1 \ldots x_{2m}$, $u = u_1 \ldots u_m$, $v = v_1 \ldots v_m$, and $w = w_1 \ldots w_m$ with $u_i = x_{2i-1}$, $v_i = x_{2i}$, and $w_i = v_i \oplus u_i$. (Recall that $a \oplus b = 1$ iff $a \neq b$.) This solution apparently does not generalize. A general solution to distribute x over $m+k$ elements such that any group of m elements determines x can be given as follows: Compute the least integer $y \geq x^{1/m}$. Let p_i be the ith prime, with $p_1 = 2$. Distribute x over $u_1, \ldots, u_{m+k}$, where $u_i \equiv x \bmod p_i^{\alpha(i)}$, with $\alpha(i) = \lceil y \log_{p_i} 2 \rceil$. Using the Chinese remainder theorem we find that we can reconstruct x from any subset of m elements u_i. Source: A. Shamir, *Comm. ACM*, 22:11(1979), 612–613; M.O. Rabin, *J. ACM*, 36:2(1989), 335–348.

2.2.11. [26] Show that there are strings x, y, z such that $C(x|y) + C(x|z) > C(x) + C(x|y, z) + O(1)$. For convenience prove this first for strings of the same length n; but it also holds for some strings x, y, z with $l(x) = \log n$ and $l(y) = l(z) = n$. *Comments.* This is a counterintuitive result. Hint: prove there are pairwise random strings x, y, z such that each string results from $\oplus$-ing the other two.

2.2.12. [18] Let A be the set of binary strings of length n. An element x in A is *δ-random* if $\delta(x|A) \leq \delta$, where $\delta(x|A) = n - C(x|A)$ is the randomness deficiency. Show that if $x \in B \subseteq A$, then

$$\log \frac{d(A)}{d(B)} - C(B|A) \leq \delta(x|A) + O(\log n).$$

Comments. That is, no random elements of A can belong to any subset B of A that is simultaneously pure (which means that $C(B|A)$ is small) and not large (which means that $d(A)/d(B)$ is large). Source: A.N. Kolmogorov and V.A. Uspensky, *Theory Probab. Appl.*, 32(1987), 389–412.

2.2.13. [27] Let $x \in A$, with $d(A) < \infty$. Then in Section 2.2 the *randomness deficiency* of x *relative* to A is defined as $\delta(x|A) = l(d(A)) - C(x|A)$. (Here $C(x|A)$ is defined as $C(x|\chi)$ with χ the characteristic sequence of A and $l(\chi) < \infty$.) If $\delta(x|A)$ is large, this means that there is a description of x with the help of A that is considerably shorter than just giving x's serial number in A. Clearly, the randomness deficiency of x with respect to sets A and B can be vastly different. But then it is natural to ask whether there exist *absolutely nonrandom* objects, objects having large randomness deficiency with respect to any appropriate set.

Prove the following: Let a and b be arbitrary constants; for every sufficiently large n, there exists a binary string x of length n such that $\delta(x|A) \geq b \log n$ for any set A containing x for which $C(A) \leq a \log n$.

Comments. Source: A.K. Shen, *Soviet Math. Dokl.*, 28(1983), 295–299. Compare with Kamae's theorem, Exercise 2.7.5. Let us give some interpretation of such results bearing on statistical inference. Given an experimental result, the statistician wants to infer a statistical hypothesis under which the result is *typical*. Mathematically, given x we want to find a simple set A that contains x as a typical element. The above shows that there are outcomes x such that *no* simple statistical model of the kind described is possible. The question remains whether such objects occur in the real world.

2.2.14. [31] Consider two complexity measures for infinite binary sequences ω. Let $C_\infty(\omega)$ be the minimal length of a program p such that $p(n) = \omega_{1:n}$ for all sufficiently large n. Let $\hat{C}_\infty(\omega)$ be defined as $\limsup_{n\to\infty} C(\omega_{1:n}|n)$. Prove that $C_\infty(\omega) \leq 2\hat{C}_\infty(\omega) + O(1)$, and that this bound is tight (the constant 2 cannot be replaced by a smaller one).

Comments. Source: B. Durand, A.K. Shen, N.K. Vereshchagin. *Theoret. Comput. Sci.*, 171(2001), 47–58.

2.2.15. [37] Consider $C_{\lim}(x) = \min\{l(p) : p(n) = x$ for all but finitely many $n\}$ and $C_{\limsup}(x) = \min\{m :$ for all but finitely many n there exists a p with $l(p) \leq m$ and $p(n) = x\}$. Let $C'(x)$ denote the plain Kolmogorov complexity relativized to $0'$ (that is, the program is allowed to ask an oracle whether a given Turing machine terminates on given input).

(a) Prove that $C_{\lim}(x) = C'(x) + O(1)$.

(b) Prove that $C_{\limsup}(x) = C'(x) + O(1)$.

Comments. Source: N.K. Vereshchagin *Theoret. Comput. Sci.*, 271(2002), 59–67. Item (b) is the more difficult one; Item (a) is attributed to An.A. Muchnik, S.Y. Positselsky.

2.3 C as an Integer Function

We consider C as an integer function $C : \mathcal{N} \rightarrow \mathcal{N}$, and study its behavior, Figure 2.1. First we observe that Theorem 2.1.2 gives an *upper bound* on C: there exists a constant c such that for all x in $\mathcal{N}$ we have $C(x) \leq l(x) + c$, and by Theorem 2.2.1 this estimate is almost exact for the majority of x's. This is a computable monotonic increasing upper bound that grows to infinity. It is also the least such upper bound. It turns out that the greatest monotonic nondecreasing lower bound also grows to infinity but does so incomputably slowly.

Theorem 2.3.1 (i) *The function $C(x)$ is unbounded.*

(ii) *Define a function m by $m(x) = \min\{C(y) : y \geq x\}$. That is, m is the greatest monotonic increasing function bounding C from below. The function $m(x)$ is unbounded.*

(iii) *For any partial recursive function $\phi(x)$ that goes monotonically to infinity from some x_0 onward, we have $m(x) < \phi(x)$ except for finitely many x. In other words, although $m(x)$ goes to infinity, it does so more slowly than any unbounded partial recursive function.*

Proof. (i) This follows immediately from (ii).

(ii) For each i there is a least x_i such that for all $x > x_i$, the smallest program p printing x has length greater than or equal to i. This follows immediately from the fact that there are only a finite number of programs of each length i. Clearly, for all i we have $x_{i+1} \geq x_i$. Now observe that the function m has the property that $m(x) = i$ for $x_i < x \leq x_{i+1}$.

(iii) Assume the contrary: there is a monotonic nondecreasing unbounded partial recursive function $\phi(x) \leq m(x)$ for infinitely many x. The domain $A = \{x : \phi(x) < \infty\}$ of ϕ is an infinite recursively enumerable set. By Lemma 1.7.4, A contains an infinite recursive subset B. Define

$$\psi(x) = \begin{cases} \phi(x) & \text{for } x \in B, \\ \phi(y) & \text{with } y = \max\{z : z \in B, z < x\}, \text{ otherwise.} \end{cases}$$

This ψ is total recursive and goes monotonically to infinity, and $\psi(x) \leq m(x)$ for infinitely many x.

Now define $M(a) = \max\{x : C(x) \leq a\}$. Then, $M(a) + 1 = \min\{x : m(x) > a\}$. It is easy to verify that

$$\max\{x : \psi(x) \leq a + 1\} \geq \min\{x : m(x) > a\} > M(a),$$

for infinitely many a's, and the function $F(a) = \max\{x : \psi(x) \leq a + 1\}$ is obviously total recursive. Therefore, $F(a) > M(a)$ for infinitely many a's. In other words, $C(F(a)) > a$ for infinitely many a's. But by Theorem 2.1.1,

$$C(F(a)) \leq C_F(F(a)) + O(1) \leq l(a) + O(1).$$

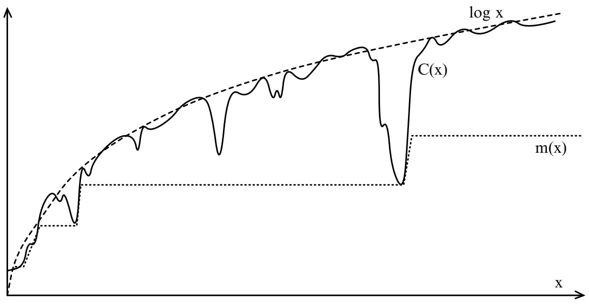

FIGURE 2.1. The graph of the integer function $C(x)$

This implies that there exists a constant c such that $l(a) + c \geq a$ for infinitely many a, which is impossible. $\square$

Notice that Items (ii) and (iii) of Theorem 2.3.1 do not hold for the length-conditional complexity $C(x|l(x))$. Namely, although $C(x|l(x))$ is unbounded, it drops infinitely often to constant level. In other words, there is no unbounded monotonic function that is a lower bound on $C(x|l(x))$ by Example 2.2.5. This phenomenon is further explored in the exercises.

The second cornerstone of the theory (millstone around its neck is probably more apt) is the *incomputability theorem.*

Theorem 2.3.2 *The function $C(x)$ is not recursive. Moreover, no partial recursive function $\phi(x)$ defined on an infinite set of points can coincide with $C(x)$ over the whole of its domain of definition.*

Proof. This proof is related to that of Theorem 2.3.1, Item (iii). We prove that there is no partial recursive ϕ as in the statement of the theorem. Every infinite recursively enumerable set contains an infinite recursive subset, Lemma 1.7.4. Select an infinite recursive subset A in the domain of definition of ϕ. The function $\psi(m) = \min\{x : C(x) \geq m, x \in A\}$ is (total) recursive (since $C(x) = \phi(x)$ on A), and takes arbitrarily large values, Theorem 2.3.1. Also, by definition of ψ, we have $C(\psi(m)) \geq m$. On the other hand, $C(\psi(m)) \leq C_\psi(\psi(m)) + c_\psi$ by definition of C, and obviously $C_\psi(\psi(m)) \leq l(m)$. Hence, $m \leq \log m$ up to a constant independent of m, which is false from some m onward. $\square$

That was the bad news; the good news is that we can approximate $C(x)$.

Theorem 2.3.3 *There is a total recursive function $\phi(t, x)$, monotonic decreasing in t, such that* $\lim_{t\to\infty} \phi(t, x) = C(x)$.

Proof. We define $\phi(t,x)$ as follows: For each x, we know that the shortest program for x has length at most $l(x)+c$, Theorem 2.1.2. Run the reference Turing machine U in the proof of Theorem 2.1.1 for t steps on *each* program p of length at most $l(x)+c$. If for any such input p the computation halts with output x, then define the value of $\phi(t,x)$ as the length of the shortest such p, otherwise equal to $l(x)+c$. Clearly, $\phi(t,x)$ is recursive, total, and monotonically nonincreasing with t (for all x, $\phi(t',x) \leq \phi(t,x)$ if $t' > t$). The limit exists, since for each x there exists a t such that U halts with output x after computing t steps starting with input p with $l(p) = C(x)$. □

One cannot decide, given x and t, whether $\phi(t,x) = C(x)$. Since $\phi(t,x)$ is nondecreasing and goes to the limit $C(x)$ for $t \rightarrow \infty$, if there were a decision procedure to test $\phi(t,x) = C(x)$, given x and t, then we could compute $C(x)$. But Theorem 2.3.2 tells us that C is not recursive.

Let $g_1, g_2, \ldots$ be a sequence of functions. We call f the *limit* of this sequence if $f(x) = \lim_{t\rightarrow\infty} g_t(x)$ for all x. The limit is *recursively uniform* if for every rational $\epsilon > 0$ there exists a $t(\epsilon)$, where t is a total recursive function, such that $|f(x) - g_{t(\epsilon)}(x)| \leq \epsilon$, for all x. Let the sequence of one-argument functions $\psi_1, \psi_2, \ldots$ be defined by $\psi_t(x) = \phi(t,x)$, for each t for all x. Clearly, C is the limit of the sequence of ψ's. However, by Theorem 2.3.2, the limit is not recursively uniform. In fact, by the halting problem in Section 1.7, for each $\epsilon > 0$ and $t > 0$ there exist infinitely many x such that $|C(x) - \psi_t(x)| > \epsilon$. This means that for each $\epsilon > 0$, for each t there are many x's such that our estimate $\phi(t,x)$ overestimates $C(x)$ by an *error* of at least ϵ.

We describe some other characteristics of the function C.

Continuity: The function C is continuous in the sense that there is a constant c such that $|C(x) - C(x \pm h)| \leq 2l(h) + c$ for all x and h. (Hint: given a program that computes x we can change it into another program that adds (or subtracts) h from the output.)

Logarithmic: The function $C(x)$ mostly hugs $\log x$. It is bounded above by $\log x + c$, Theorem 2.1.2, page 108. On the other hand, by Theorem 2.2.1, page 117, for each constant k, the number of x of length n (about $\log x$) such that $C(x) < \log x - k$ is at most 2^{n-k}.

Fluctuation: The function $C(x)$ fluctuates rapidly. Namely, for each x there exist two integers x_1, x_2 within distance $\sqrt{x}$ of x (that is, $|x - x_i| \leq \sqrt{x}$ for $i = 1,2$) such that $C(x_1) \geq l(x)/2 - c$ and $C(x_2) \leq l(x)/2 + c$. (Hint: change the low-order half of the bits of x to some incompressible string to obtain x_1, and change these bits to a very compressible string (such as all zeros) to obtain x_2.) Therefore, if x is incompressible with $C(x) = l(x) - O(1)$, then there is an x_2 nearby where $C(x_2)$ equals about $C(x)/2$, and if x is compressible with $C(x) = o(l(x))$, then there is an x_1 nearby where

$C(x_1)$ equals about $l(x)/2$. These facts imply many fluctuations within small intervals because, for instance, $C(x)$, $C(x + \log x)$, $C(x+\sqrt{x})$, $C(cx)$, $C(x^2)$, and $C(2^x)$ all have about the same value.

Long high-complexity runs: For each c there is a d such that there are no runs of d consecutive c-incompressible numbers. However, conversely, for each d there is a c such that there are runs of d consecutive c-incompressible numbers. (Hint: for the nonexistence part use numbers x of the form $i2^j$ for which $C(i2^j) \leq l(i) + l(j) + c < l(i2^j) - d$; for the existence part use the continuity property and the nearly logarithmic property above.)

Example 2.3.1 It is not difficult to see that Theorems 2.3.1, Item (i), 2.3.2, and 2.3.3, Theorem 2.1.2, and the above properties hold for the length-conditional complexity measure $C(x|l(x))$. By the existence of n-strings, Example 2.2.5, the greatest monotonic lower bound on $C(x|l(x))$ is a fixed constant, and therefore Items (ii) and (iii) of Theorem 2.3.1 do not hold for this complexity measure. Theorems 2.1.1, 2.2.1 are already proved for $C(x|l(x))$ in their original versions. Namely, either they were proved for the conditional complexity in general, or the proof goes through as given for the length-conditional complexity. Thus, the general contour of the graph of $C(x|l(x))$ looks *very roughly* similar to that of $C(x)$, except that there are dips below a fixed constant infinitely often, Figure 2.2.

Let us make an estimate of how often the dips occur. Consider the n-strings of Example 2.2.5. For each integer n there is an extension of the corresponding binary string with $n - l(n)$ many 0's such that the resulting string x has complexity $C(x|l(x)) \leq c$ for a fixed constant c. It is easy to see that $\log x \approx n$, and that for all $n' < n$ the corresponding x' is less than x. Hence, the number of $x' < x$ such that $C(x'|l(x')) \leq c$ is at least $\log x$. ◇

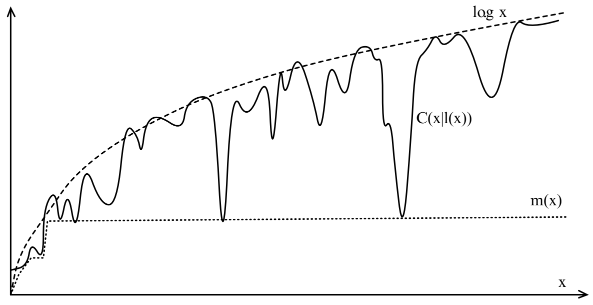

FIGURE 2.2. The graph of the integer function $C(x|l(x))$

Exercises

2.3.1. [15] Let $\phi(t,x)$ be a recursive function and $\lim_{t\rightarrow\infty}\phi(t,x) = C(x)$, for all x. For each t define $\psi_t(x) = \phi(t,x)$ for all x. Then C is the *limit* of the sequence of functions $\psi_1, \psi_2, \ldots$. Show that for each *error* ϵ and all t there are infinitely many x such that $\psi_t(x) - C(x) > \epsilon$.

Comments. $C(x)$ is the *uniform limit* of the approximation above if for each $\epsilon > 0$, there exists a t such that for all x, $\psi_t(x) - C(x) \leq \epsilon$. Item (a) implies that $C(x)$ is not the uniform limit of the above sequence of functions.

2.3.2. [23] Let $\phi_1, \phi_2, \ldots$ be the effective enumeration of partial recursive functions in Section 1.7. Define the *uniform complexity* of a finite string x of length n with respect to ϕ (occurring in the above enumeration) as $C_\phi(x;n) = \min\{l(p) : \phi(m,p) = x_{1:m} \text{ for all } m \leq n\}$ if such a p exists, and ∞ otherwise. We can prove an invariance theorem to the effect that there exists a universal partial recursive function ϕ_0 such that for all ϕ there is a constant c such that for all x, n we have $C_{\phi_0}(x;n) \leq C_\phi(x;n) + c$. We choose a reference universal function ϕ_0 and define the *uniform Kolmogorov complexity* as $C(x;n) = C_{\phi_0}(x;n)$.

(a) Show that for all finite binary strings x we have $C(x) \geq C(x;l(x)) \geq C(x|l(x))$ up to additive constants independent of x.

(b) Prove Theorems 2.1.1 to 2.3.3, with $C(x)$ replaced by $C(x;l(x))$.

(c) Show that in contrast to the measure $C(x|l(x))$, no constant c exists such that $C(x;l(x)) \leq c$ for all n-strings (Example 2.2.5).

(d) Show that in contrast to $C(x|l(x))$, the uniform complexity is *monotonic in the prefixes*: if $m \leq n$, then $C(x_{1:m};m) \leq C(x_{1:n};n)$, for all x.

(e) Show that there exists an infinite binary sequence ω and a constant c such that for infinitely many n, $C(\omega_{1:n};n) - C(\omega_{1:n}|n) > \log n - c$.

Comments. Item (b) shows that the uniform Kolmogorov complexity satisfies the major properties of the plain Kolmogorov complexity. Items (c) and (d) show that at least two of the objections to the length-conditional measure $C(x|l(x))$ do not hold for the uniform complexity $C(x;l(x))$. Hint for Item (c): this is implied by the proof of Theorem 2.3.1 and Item (a). Item (e) shows as strong a divergence between the measures concerned as one could possibly expect. Source: D.W. Loveland, *Inform. Contr.*, 15(1969), 510–526.

2.3.3. [27] Let BB' be a variant of the busy beaver function defined in Exercise 1.7.19, page 45, where $BB'(n)$ is defined as the maximal number of steps in a halting computation of the reference universal Turing machine when started on an n-bit input.

Show that $C(BB'(n)) = n + O(\log n)$. Use this to provide an alternative proof for Theorem 2.3.1, Item (iii).

Comments. Hint: Knowing n and the index $j \leq 2^n$ of the input that achieves $BB'(n)$, we can compute $BB'(n)$. Hence, $C(BB'(n) \mid n) \leq n + O(1)$. On the other hand, Knowing n and $BB'(n)$, we can run all programs of n bits for at most $BB'(n)$ steps; the programs that have not halted after $BB'(n)$ steps will never halt. This resolves the halting problem for all programs of n bits, and yields the halting sequence $\chi_1 \ldots \chi_{2^n}$ for the first 2^n programs. By an application of the later Theorem 2.7.2, known as Barzdins's lemma, Item (ii), we conclude that $C(BB'(n), n) \geq C(\chi_1 \ldots \chi_{2^n}) - O(1) \geq n - O(1)$.

2.3.4. • [35] Let ω be an infinite binary string. We call ω *recursive* if there exists a recursive function ϕ such that $\phi(i) = \omega_i$ for all $i > 0$. Prove the following:

(a) If ω is recursive, then there is a constant c such that for all n,

$$C(\omega_{1:n}; n) < c,$$
$$C(\omega_{1:n}|n) < c,$$
$$C(\omega_{1:n}) - C(n) < c.$$

This is easy. The converses also hold but are less easy to show. They follow from Items (b), (e), and (f).

(b) For each constant c, there are only finitely many ω such that for all n, $C(\omega_{1:n}; n) \leq c$, and each such ω is recursive.

(c) For each constant c, there are only finitely many ω such that for infinitely many n, $C(\omega_{1:n}; n) \leq c$, and each such ω is recursive.

(d) There exists a constant c such that the set of infinite ω that satisfy $C(\omega_{1:n}|n) \leq c$ for infinitely many n, has the cardinality of the continuum.

(e) For each constant c, there are only finitely many ω such that for all n, $C(\omega_{1:n}|n) \leq c$, and each such ω is recursive.

(f) For each constant c, there are only finitely many ω with $C(\omega_{1:n}) \leq C(n) + c$ for all n, and each such ω is recursive.

(g) For each constant c, there are only finitely many ω with $C(\omega_{1:n}) \leq l(n) + c$ for all n, and each such ω is recursive.

(h) There exist nonrecursive ω for which there exists a constant c such that $C(\omega_{1:n}) \leq C(n) + c$ for infinitely many n.

Comments. Clearly Item (c) implies Item (b). In Item (d) conclude that not all such ω are recursive. In particular, the analogue of Item (c) for $C(\omega_{1:n}|n)$ does not hold. Namely, there exist nonrecursive ω for which there exists a constant c such that for infinitely many n we have $C(\omega_{1:n}|n) \leq c$. Hint for Item (d): exhibit a one-to-one coding of subsets of $\mathcal{N}$ into the set of infinite binary strings of which infinitely many prefixes are n-strings—in the sense of Example 2.2.5. Item (e) means that in contrast to the differences between the measures $C(\cdot; l(\cdot))$ and $C(\cdot|l(\cdot))$ exposed by the contrast between Items (c) and (d), Item (b) holds also for $C(\cdot|l(\cdot))$. Items (f) and (g) show a complexity gap, because $C(n)$ can be much lower than $l(n)$. Hint for Item (h): use Item (d). Source: for Items (b) through (e), and (h), D.W. Loveland, *Inform. Contr.*, 15(1969), 510–526. Loveland attributes Item (e) to A.R. Meyer. The equivalence between bounded length-conditional complexity and bounded uniform complexity for prefixes of infinite strings is stated by A.K. Zvonkin and L.A. Levin, *Russ. Math. Surv.*, 25:6(1970), 83–124. Source of Items (f) and (g) is G.J. Chaitin, *Theoret. Comput. Sci.*, 2(1976), 45–48. For the prefix complexity K introduced in Chapter 3, there are nonrecursive ω such that $K(\omega_{1:n}) \leq K(n) + O(1)$ for all n by a result of R.M. Solovay in Exercise 3.6.9 on page 231.

2.3.5. [HM35] We want to show in some precise sense that the real line is computationally a fractal. (Actually, one is probably most interested in Item (a), which can be proved easily and elementarily from the following definition.) The required framework is as follows: Each infinite binary sequence $\omega = \omega_1\omega_2\ldots$ corresponds to a real number $0 \leq 0.\omega < 1$. Define the normalized complexity $Cn(\omega) = \lim_{n\to\infty} C(\omega_{1:n})/n$. If the limit does not exist, we set $Cn(\omega)$ to half the sum of the upper and lower limits.

(a) Show that for all real ω in $[0, 1)$, for every $\epsilon > 0$ and all real r, $0 \leq r \leq 1$, there exist real ζ in $[0, 1)$ such that $|\omega-\zeta| < \epsilon$ and $Cn(\zeta) = r$. (For each real r, $0 \leq r \leq 1$, the set of ω's with $Cn(\omega) = r$ is dense on the real line $[0, 1)$.)

(b) Show that for all real ω, all rational r and s, we have $Cn(r\omega + s) = Cn(\omega)$ (both ω and $r\omega + s$ in $[0, 1)$). Similarly, show that $Cn(f(\omega)) = Cn(\omega)$ for all recursive functions f.

B. Mandelbrot defined a set to be a *fractal* if its Hausdorff dimension is greater than its topological dimension [B. Mandelbrot, *The Fractal Geometry of Nature*, W.H. Freeman, 1983; for definitions of the dimensions see W. Hurewicz and H. Wallman, *Dimension Theory*, Princeton Univ. Press, 1974].

(c) Show that for any real numbers $0 \leq a < b \leq 1$, the Hausdorff dimension of the set $\{(\omega, Cn(\omega))\} \bigcap([0, 1) \times [a, b])$ is $1 + b$.

(d) Show that the set $G = \{(\omega, Cn(\omega)) : \omega \in [0, 1)\}$ has Hausdorff dimension 2 and topological dimension 1. (That is, G is a fractal.)

Comments. Source: J.-Y. Cai and J. Hartmanis, *J. Comput. System Sci.*, 49:3(1994), 605–619. Other relationships among the Hausdorff dimension, Lutz's constructive dimension, and Kolmogorov complexity have been investigated by L. Staiger in [*Inform. Comput.*, 102(1993), 159-194; *Theor. Comput. Syst.* 31(1998), 215-229], B.Ya. Ryabko in [*J.*

Complexity, 10(1994) 281–295]; J.H. Lutz in [*Proc. 27th Int. Colloq. Aut. Lang. Prog.*, 2000, pp. 902–913; *Inform. Comput.*, 187(2003), pp. 49-79; *SIAM J. Comput.* 32(2003), 1236-1259], and E. Mayordomo in [*Inform. Process. Lett.*, 84:1(2002), 247–356].

2.3.6. [M34] To investigate repeating patterns in the graph of $C(x)$ we define the notion of a 'shape match.' Every function from the integers to the integers is a shape. A *shape f matches* the graph of C at j with *span* c if for all x with $j-c \leq x \leq j+c$ we have $C(x) = C(j) + f(x-j)$.

(a) Show that every matching shape has $f(0) = 0$. Thus, a matching shape is a template of which we align the center $f(0)$ with j to see to what extent it matches C's graph around the point of interest. We wish to investigate shapes that can be made to match arbitrarily far in each direction.

(b) A shape f is *recurrent* if for all c there exists a j such that f matches the graph of C at j with span c. Show that there exists a recurrent shape.

(c) Show that there exists a constant c such that there are no runs $C(n) = C(n+1) = \cdots = C(n+c)$ for any n.

(d) Prove that no recurrent shape is a recursive function.

Comments. The notion of 'shape match' is different from that of 'following the shape' in Definition 5.5.8 on page 407. Hints: for Item (b) use König's infinity lemma. Item (c) means that the graph of C has no arbitrarily long flat spots. For Item (c), prove for sufficiently large c that for each integer i, for all n with $C(n) = i$, the run $C(n), C(n+1), \ldots, C(n+c)$ contains an element less than i. For Item (d) use a case analysis, and use in one case the proof of Item (c) and in the other cases the recursion theorem, Exercises 1.7.20, page 46. Source: H.P. Katseff and M. Sipser, *Theoret. Comput. Sci.*, 15(1981), 291–309.

2.4 Random Finite Sequences

One can consider those objects as nonrandom in which one can find sufficiently many regularities. In other words, we would like to identify incompressibility with randomness. This is proper if the sequences that are incompressible can be shown to possess the various properties of randomness (stochasticity) known from the theory of probability. That this is possible is the substance of the celebrated theory developed by the Swedish mathematician Per Martin-Löf.

There are many properties known that probability theory attributes to random objects. To give an example, consider sequences of n tosses with a fair coin. Each sequence of n zeros and ones is equiprobable as an outcome: its probability is 2^{-n}. If such a sequence is to be random in the sense of a proposed new definition, then the number of ones in x

should be near to $\frac{1}{2}n$, the number of occurrences of blocks 00 should be close to $\frac{1}{4}n$, and so on.

It is not difficult to show that each such single property separately holds for all incompressible binary strings. But we want to demonstrate that incompressibility implies all conceivable effectively testable properties of randomness (both the known ones and the as yet unknown ones). In this way, the various theorems in probability theory about random sequences carry over automatically to incompressible sequences.

In the case of finite strings we cannot hope to distinguish sharply between random and nonrandom strings. For instance, considering the set of binary strings of a fixed length, it would not be natural to fix an m and call a string with m zeros random and a string with $m+1$ zeros nonrandom.

Let us borrow some ideas from statistics. We are given a certain sample space S with an associated distribution P. Given an element x of the sample space, we want to test the hypothesis "x is a typical outcome." Practically speaking, the property of being typical is the property of belonging to any reasonable majority. In choosing an object at random, we have confidence that this object will fall precisely in the intersection of all such majorities. The latter condition we identify with x being random.

To ascertain whether a given element of the sample space belongs to a particular reasonable majority, we introduce the notion of a test. Generally, a test is given by a prescription that for every level of significance ϵ, tells us for what elements x of S the hypothesis "x belongs to majority M in S" should be rejected, where $\epsilon = 1 - P(M)$. Taking $\epsilon = 2^{-m}$, $m = 1, 2, \ldots$, we achieve this by saying that we have a description of the set $V \subseteq \mathcal{N} \times S$ of nested *critical regions*

$$V_m = \{x : (m, x) \in V\},$$
$$V_m \supseteq V_{m+1}, \qquad m = 1, 2, \ldots,$$

while the condition that V_m be a critical region on the *significance level* $\epsilon = 2^{-m}$ amounts to requiring, for all n,

$$\sum_x \{P(x|l(x) = n) : x \in V_m\} \leq \epsilon.$$

The complement of a critical region V_m is called the $(1-\epsilon)$ *confidence interval.* If $x \in V_m$, then the hypothesis "x belongs to majority M," and therefore the stronger hypothesis "x is random," is rejected with significance level ϵ. We can say that x fails the test at the level of critical region V_m.

Example 2.4.1 A string $x_1 x_2 \ldots x_n$ with many initial zeros is not very random. We can test this aspect as follows. The special test V has critical regions

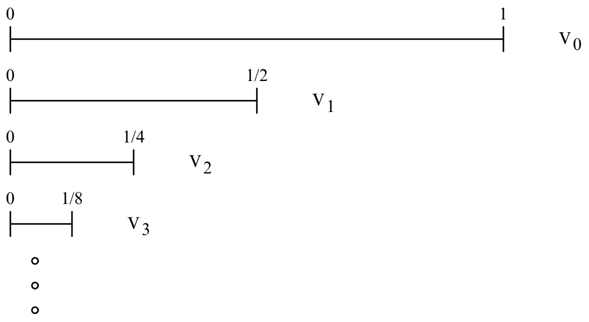

FIGURE 2.3. Test of Example 2.4.1

$V_1, V_2, \ldots$. Consider $x = 0.x_1x_2 \ldots x_n$ as a rational number, and each critical region as a half-open interval $V_m = [0, 2^{-m})$ in $[0,1)$, $m = 1, 2, \ldots$. Then the subsequent critical regions test the hypothesis "x is random" by considering the subsequent digits in the binary expansion of x. We reject the hypothesis on the significance level $\epsilon = 2^{-m}$ provided $x_1 = x_2 = \cdots = x_m = 0$, Figure 2.3. ◇

Example 2.4.2 Another test for randomness of finite binary strings rejects when the relative frequency of ones differs too much from $\frac{1}{2}$. This particular test can be implemented by rejecting the hypothesis of randomness of $x = x_1x_2 \ldots x_n$ at level $\epsilon = 2^{-m}$ provided $|2f_n - n| > g(n,m)$, where $f_n = \sum_{i=1}^{n} x_i$, and $g(n,m)$ is the least number determined by the requirement that the number of binary strings x of length n for which this inequality holds be at most 2^{n-m}. Thus, in this case the critical region V_m is $\{x \in \{0,1\}^n : |2f_n - n| > g(n,m)\}$. ◇

2.4.1 Randomness Tests

In practice, statistical tests are *effective* prescriptions such that we can compute, at each level of significance, for what strings the associated hypothesis should be rejected. It would be hard to imagine what use it would be in statistics to have tests that are not effective in the sense of computability theory (Section 1.7).

Definition 2.4.1 Let P be a recursive probability distribution on sample space $\mathcal{N}$. A total function $\delta : \mathcal{N} \to \mathcal{N}$ is a *P-test* (Martin-Löf test for randomness) if

1. δ is lower semicomputable (the set $V = \{(m,x) : \delta(x) \geq m\}$ is recursively enumerable); and
2. $\sum_x \{P(x|l(x) = n, \delta(x) \geq m\} \leq 2^{-m}$, for all n.

The critical regions associated with the common statistical tests are present in the form of the sequence $V_1 \supseteq V_2 \supseteq \cdots$, where $V_m = \{x : \delta(x) \geq m\}$, for $m \geq 1$. Nesting is ensured, since $\delta(x) \geq m+1$ implies $\delta(x) \geq m$. Each set V_m is recursively enumerable because of Item 1.

Consider the important case of the uniform distribution, defined by $L(x) = 2^{-2l(x)-1}$. The restriction of L to strings of length n is defined by $L_n(x) = 2^{-n}$ for $l(x) = n$ and 0 otherwise. (By definition, $L_n(x) = L(x|l(x) = n)$.) Then Item 2 can be rewritten as $\sum_{x \in V_m} L_n(x) \leq 2^{-m}$, which is the same as

$$d(\{x : l(x) = n,\ x \in V_m\}) \leq 2^{n-m}.$$

In this case we often speak simply of a *test*, with the uniform distribution L understood.

In Definition 2.4.1, the integer function δ is total and the set of points V of its graph is recursively enumerable. But the totality of δ implies that the recursively enumerable set V is actually recursive, and therefore we can require δ to be a total recursive function without changing the notion of P-test.

In statistical tests, membership of (m, x) in V can usually be determined in time polynomial in $l(m) + l(x)$.

Note that

$$\sum_x P(x)\delta(x) = \sum_m P\{x : \delta(x) \geq m\} \leq \sum_m 2^{-m} = 2.$$

Therefore, $\delta'(x) = \log \delta(x)$ is almost a sum-P test, Definition 4.3.8 on page 278.

Example 2.4.3 The previous test examples can be rephrased in terms of Martin-Löf tests. Let us try a more subtle example. A real number such that all bits in odd positions in its binary representation are 1's is not random with respect to the uniform distribution. To show this we need a test that detects sequences of the form $x = 1x_2 1x_4 1x_6 1x_8 \ldots$. Define a test δ by

$$\delta(x) = \max\{i : x_1 = x_3 = \cdots = x_{2i-1} = 1\},$$

and $\delta(x) = 0$ if $x_1 = 0$. For example: $\delta(01111) = 0$; $\delta(10011) = 1$; $\delta(11011) = 1$; $\delta(10100) = 2$; $\delta(11111) = 3$. To show that δ is a test we have to show that δ satisfies the definition of a test. Clearly, δ is lower semicomputable (even recursive). If $\delta(x) \geq m$ where $l(x) = n \geq 2m-1$, then there are 2^{m-1} possibilities for the $(2m-1)$-length prefix of x, and $2^{n-(2m-1)}$ possibilities for the remainder of x. Therefore, $d\{x : \delta(x) \geq m,\ l(x) = n\} \leq 2^{n-m}$. ◇

Definition 2.4.2 A universal Martin-Löf test for randomness with respect to a distribution P, a *universal P-test* for short, is a test $\delta_0(\cdot|P)$ such that for each P-test δ, there is a constant c such that for all x we have $\delta_0(x|P) \geq \delta(x) - c$.

We say that $\delta_0(\cdot|P)$ (additively) majorizes δ. Intuitively, $\delta_0(\cdot|P)$ constitutes a test for randomness that incorporates all particular tests δ in a single test. No test for randomness δ other than $\delta_0(\cdot|P)$ can discover more than a constant amount of greater deficiency of randomness in any string x. In terms of critical regions, a universal test is a test such that if a binary sequence is random with respect to that test, then it is random with respect to any conceivable test, neglecting a change in significance level. Namely, with $\delta_0(\cdot|P)$ a universal P-test, let $U = \{(m,x) : \delta_0(x|P) \geq m\}$, and for any test δ, let $V = \{(m,x) : \delta(x) \geq m\}$. Then, defining the associated critical zones as before, we obtain

$$V_{m+c} \subseteq U_m, \ m = 1, 2, \ldots,$$

where c is a constant (dependent only on U and V).

It is a major result that there exists a universal P-test. The proof goes by first showing that the set of all tests is enumerable. This involves the first example of a type of construction we shall use over and over again in different contexts in Chapters 2, 3, and 4. The idea is as follows:

Lemma 2.4.1 *We can effectively enumerate all P-tests.*

Proof. We start with the standard enumeration $\phi_1, \phi_2, \ldots$ of partial recursive functions from $\mathcal{N}$ into $\mathcal{N} \times \mathcal{N}$, and turn this into an enumeration $\delta_1, \delta_2, \ldots$ of all and only P-tests. The list $\phi_1, \phi_2, \ldots$ enumerates all and only recursively enumerable sets of pairs of integers as $\{\phi_i(x) : x \geq 1\}$ for $i = 1, 2, \ldots$. In particular, for any P-test δ, the set $\{(m,x) : \delta(x) \geq m\}$ occurs in this list. The *only* thing we have to do is to eliminate those ϕ_i whose range does not correspond to a P-test.

First, we effectively modify each ϕ (we drop the subscript for convenience) to a function ψ such that range ϕ equals range ψ, and ψ has the special property that if $\psi(n)$ is defined, then $\psi(1), \psi(2), \ldots, \psi(n-1)$ are also defined. This can be done by dovetailing the computations of ϕ on the different arguments: in the first phase, do one step of the computation of $\phi(1)$; in the second phase, do the second step of the computation of $\phi(1)$ and the first step of the computation of $\phi(2)$. In general, in the nth phase we execute the n_1th step of the computation of $\phi(n_2)$, for all n_1, n_2 satisfying $n_1 + n_2 = n$. We now define ψ as follows. If the first computation that halts is that of $\phi(i)$, then set $\psi(1) := \phi(i)$. If the second computation that halts is that of $\phi(j)$, then set $\psi(2) := \phi(j)$, and so on.

Secondly, use each ψ to construct a test δ by approximation from below. In the algorithm, at each stage of the computation the local variable array $\delta(1 : \infty)$ contains the current approximation to the list of function values $\delta(1), \delta(2), \ldots$. This is doable because the nonzero part of the approximation is always finite.

Step 1. Initialize δ by setting $\delta(x) := 0$ for all x; and set $i := 0$. **{**If the range of ψ is empty, then this assignment will not be changed in the remainder of the procedure, that is, δ stays identically zero and it is trivially a test**}**

Step 2. Set $i := i + 1$; compute $\psi(i)$ and let its value be (m, x).

Step 3. **If** $\delta(x) \geq m$ **then go to** Step 2 **else** set $\delta(x) := m$.

Step 4. **If** $\sum\{P(y|l(y) = l(x)) : \delta(y) \geq k\} > 2^{-k}$ for some k, $k = 1, \ldots, m$ **{**since P is a recursive function we can effectively test whether the new value of $\delta(x)$ violates Definition 2.4.1 on page 135**}** **then** set $\delta(x) := 0$ and terminate **{**the computation of δ is finished**}** **else go to** Step 2.

With P the uniform distribution, for $i = 1$ the conditional in Step 4 simplifies to $m > l(x)$. In case the range of ψ is already a test, then the algorithm never finishes but forever approximates δ from below. If ψ diverges for some argument then the computation goes on forever and does not change δ any more. The resulting δ is a lower semicomputable test. If the range of ψ is not a test, then at some point the conditional in Step 4 is violated and the approximation of δ terminates. The resulting δ is a test, even a recursive one. Executing this procedure on all functions in the list $\phi_1, \phi_2, \ldots$, we obtain an effective enumeration $\delta_1, \delta_2, \ldots$ of all P-tests (and only P-tests). We are now in a position to define a universal P-test. $\square$

Theorem 2.4.1 *Let $\delta_1, \delta_2, \ldots$ be an enumeration of the above P-tests. Then $\delta_0(x|P) = \max\{\delta_y(x) - y : y \geq 1\}$ is a universal P-test.*

Proof. Note first that $\delta_0(\cdot|P)$ is a total function on $\mathcal{N}$ because of Item 2 in Definition 2.4.1, page 135.

(1) The enumeration $\delta_1, \delta_2, \ldots$ in Lemma 2.4.1 yields an enumeration of recursively enumerable sets:

$$\{(m, x) : \delta_1(x) \geq m\}, \ \{(m, x) : \delta_2(x) \geq m\}, \ldots .$$

Therefore, $V = \{(m, x) : \delta_0(x|P) \geq m\}$ is recursively enumerable.

(2) Let us verify that the critical regions are small enough: for each n,

$$\sum_{l(x)=n} \{P(x|l(x) = n) : \delta_0(x|P) \geq m\}$$

$$\leq \sum_{y=1}^{\infty} \sum_{l(x)=n} \{P(x|l(x) = n) : \delta_y(x) \geq m + y\}$$

$$\leq \sum_{y=1}^{\infty} 2^{-m-y} = 2^{-m}.$$

(3) By its definition, $\delta_0(\cdot|P)$ majorizes each δ additively. Hence, it is universal. □

By definition of $\delta_0(\cdot|P)$ as a universal P-test, any particular P-test δ can discover at most a constant amount more regularity in a sequence x than does $\delta_0(\cdot|P)$, in the sense that for each δ_y we have $\delta_y(x) \leq \delta_0(x|P) + y$ for all x.

For any two universal P-tests $\delta_0(\cdot|P)$ and $\delta'_0(\cdot|P)$, there is a constant $c \geq 0$ such that for all x we have $|\delta_0(x|P) - \delta'_0(x|P)| \leq c$.

2.4.2 Explicit Universal Randomness Test

We started out with the objective to establish in what sense incompressible strings may be called random. In Section 2.2.1 we considered the randomness deficiency $\delta(x|A)$ of a string x relative to a finite set A. With A the set of strings of length n and $x \in A$ we find that $\delta(x|A) = \delta(x|n) = n - C(x|n)$.

Theorem 2.4.2 *The function $\delta_0(x|L) = l(x) - C(x|l(x)) - 1$ is a universal L-test with L the uniform distribution.*

Proof. (1) We first show that $f(x) = \delta_0(x|L)$ is a test with respect to the uniform distribution. The set $\{(m, x) : f(x) \geq m\}$ is recursively enumerable by Theorem 2.3.3.

(2) We verify the condition on the critical regions. Since the number of x's with $C(x|l(x)) \leq l(x) - m - 1$ cannot exceed the number of programs of length at most $l(x) - m - 1$, we have $d(\{x : f(x) \geq m\}) \leq 2^{l(x)-m} - 1$.

(3) We show that for each test δ, there is a constant c such that $f(x) \geq \delta(x) - c$. The main idea is to bound $C(x|l(x))$ by exhibiting a description of x, given $l(x)$. Fix x. Let the set A be defined as

$$A = \{z : \delta(z) \geq \delta(x), l(z) = l(x)\}.$$

We have defined A such that $x \in A$ and $d(A) \leq 2^{l(x)-\delta(x)}$. Let $\delta = \delta_y$ in the standard enumeration $\delta_1, \delta_2, \ldots$ of tests. Given y, $l(x)$, and $\delta(x)$, we have an algorithm to enumerate all elements of A. Together with the index j of x in enumeration order of A, this suffices to find x. We pad the standard binary representation of j with nonsignificant zeros to a string $s = 00\ldots0j$ of length $l(x) - \delta(x)$. This is possible since $l(s) \geq l(d(A))$. The purpose of changing j to s is that now the number $\delta(x)$ can be deduced from $l(s)$ and $l(x)$. In particular, there is a Turing machine that computes x from input $\bar{y}s$, when $l(x)$ is given for free. Consequently, by Theorem 2.1.1, $C(x|l(x)) \leq l(x) - \delta(x) + 2l(y) + 1$. Since y is a constant depending only on δ, we can set $c = 2l(y) + 2$. □

Definition 2.4.3 Let us fix $\delta_0(x|L) = l(x) - C(x|l(x)) - 1$ as the *reference universal test* with respect to the *uniform distribution* L. A string x is called *c-random* if $\delta_0(x|L) \leq c$.

Randomness of a string is related to its incompressibility. It is easy to see that

$$C(x|l(x)) \leq C(x) \leq C(x|l(x)) + 2C(l(x) - C(x|l(x))),$$

up to fixed additive constants. (We can reconstruct $l(x)$ from the length of a shortest program p for x and the quantity $l(x) - l(p)$.) This makes $C(x)$ and $C(x|l(x))$ about equal for the special case of x being incompressible. (However, for compressible x, such as $x = 0^n$, the difference between $C(x)$ and $C(x|n)$ can rise to logarithmic in n.) Together with Theorem 2.4.2, this provides the relation between the outcome of the reference universal L-test and incompressibility. Fix a constant c. If string x is c-incompressible, then $\delta_0(x|L) \leq c'$, where c' is a constant depending on c but not on x. Similarly, if $\delta_0(x|L) \leq c$, then x is c'-incompressible, where c' is a constant depending on c but not on x. Roughly, x is *random*, or *incompressible*, if $l(x) - C(x|l(x))$ is small with respect to $l(x)$.

Example 2.4.4 It is possible to directly demonstrate a property of random binary strings $x = x_1x_2 \ldots x_n$ related to Example 2.4.2: the number of ones, $f_n = x_1 + \cdots + x_n$, must satisfy $|2f_n - n| = O(\sqrt{n})$. Assume that x is a binary string of length n that is random in the sense of Martin-Löf. Then, by Theorem 2.4.2, x is also c-incompressible for some fixed constant c. Let $f_n = k$. The number of strings satisfying this equality is $\binom{n}{k}$. By simply giving the index of x in the lexicographic order of such strings, together with n, and the excess of ones, $d = |2k - n|$, we can give a description of x. Therefore, using a short self-delimiting program for d, we have

$$C(x|n) \leq \log \binom{n}{k} + C(d) + 2l(C(d)).$$

For x given n to be c-incompressible for a constant c, we need $C(x|n) \geq n - c$. Then,

$$n - \log \binom{n}{k} - C(d) - 2l(C(d)) \leq c,$$

which can be satisfied only if $d = O(\sqrt{n})$ (estimate the binomial coefficient by Stirling's formula, Exercise 1.5.4 on page 17). Curiously, if d given n is easily describable (for example $d = 0$ or $d = \sqrt{n}$), then x given n is not random, since it is not c-incompressible. ◇

Randomness in the sense of Martin-Löf means randomness insofar as it can be effectively certified. In other words, it is a negative definition. We look at objects through a special filter, which highlights some features but obscures others. We can perceive some qualities of nonrandomness through the limited sight of effective tests. Everything else we then call by definition random. This is a matter of a pragmatic, expedient, approach. It has no bearing on the deeper question about what properties real randomness, physically or mathematically, should have. It just tells us that an object is random as far as we will ever be able to tell, *in principle* and not just as a matter of *practicality.*

Exercises

2.4.1. [20] For a binary string x of length n, let $f(x)$ be the number of ones in x. Show that $\delta(x) = \log(2n^{-1/2}|f(x) - \frac{1}{2}n|)$ is a P-test with P the uniform measure.

Comments. Use Markov's inequality to derive that for each positive λ, the probability of $2n^{-1/2}|f(x) - \frac{1}{2}n| > \lambda$ is at most $1/\lambda$. Source: T.M. Cover, P. Gács, and R.M. Gray, *Ann. Probab.*, 17(1989), 840–865.

2.4.2. [23] Let $x_1 x_2 \ldots x_n$ be a random sequence with $C(x|n) \geq n$.

(a) Use a Martin-Löf test to show that $x_1 0 x_2 0 \ldots 0 x_n$ is not random with respect to the uniform distribution.

(b) Use a Martin-Löf test to show that the ternary sequence $y_1 y_2 \ldots y_n$ with $y_1 = x_n + x_1$ and $y_i = x_{i-1} + x_i$ for $1 < i \leq n$ is not random with respect to the uniform distribution.

Comments. Hint: in Item (b) in the y-string the blocks 02 and 20 do not occur. Extend the definition of random sequences from binary to ternary. Source: R. von Mises, *Probability, Statistics and Truth*, Dover, 1981.

2.4.3. [35] Let x be a finite binary sequence of length n with $f_j = x_1 + x_2 + \cdots + x_j$ for $1 \leq j \leq n$. Show that there exists a constant $c > 0$ such that for all $m \in \mathcal{N}$, all $\epsilon > 0$, and all x,

$$C(x|n, f_n) > \log \binom{n}{f_n} - \log(m\epsilon^4) + c$$

implies

$$\max_{m \leq j \leq n} \left| \frac{f_j}{j} - \frac{f_n}{n} \right| < \epsilon.$$

Comments. This result is called *Fine's theorem.* This is an instance of the general principle that high probability of a computable property translates into the fact that high complexity implies that property. (For infinite sequences this principle is put in a precise and rigorous form in Theorem 2.5.5.) Fine's theorem shows that for finite binary sequences with high Kolmogorov complexity, given the length and the number of ones, the fluctuations of the relative frequencies in the initial segments is small. Since we deal with finite sequences, this is called *apparent convergence.* Since virtually all finite binary strings have high complexity (Theorem 2.1.3), this explains why in a typical sequence produced by random coin throws the relative frequencies appear to converge or stabilize. Apparent convergence occurs because of, and not in spite of, the high irregularity (randomness or complexity) of a data sequence. Conversely, the failure of convergence forces the complexity to be less than maximal. Source: T.L. Fine, *IEEE Trans. Inform. Theory,* IT-16(1970), 251–257; also R. Heim, *IEEE Trans. Inform. Theory,* IT-25(1979), 557–566.

2.4.4. [36] (a) Consider a finite sequence of zeros and ones generated by independent tosses of a coin with probability p $(0 < p < 1)$ for 1. Let $x = x_1 x_2 \ldots x_n$ be a sequence of outcomes of length n, and let $f_n = x_1 + x_2 + \cdots + x_n$. The probability of such an x is $p^{f_n}(1-p)^{n-f_n}$. If p is a recursive number, then the methods in this section can be used to obtain a proper definition of finite *Bernoulli sequences*, sequences that are random for this distribution. There is, however, no reason to suppose that in physical coins p is a recursive real. This prompts another approach whereby we disregard the actual probability distribution, but follow more closely the combinatorial spirit of Kolmogorov complexity. Define a finite Bernoulli sequence as a binary sequence x of length n whose only regularities are given by f_n and n. That is, x is a Bernoulli sequence iff $C(x|n, f_n) = \log \binom{n}{f_n}$ up to a constant independent of x. Define a *Bernoulli test* as a test with the condition that the number of sequences with f_n ones and $n - f_n$ zeros in V_m be $\leq 2^{-m}\binom{n}{f_n}$ for all m, n, and f_n. Show that there exists a universal Bernoulli test δ_0. Finite Bernoulli sequences are those sequences x such that $\delta_0(x)$ is low. Show that up to a constant independent of x,

$$\delta_0(x) = \log \binom{n}{f_n} - C(x|n, f_n),$$

for all x (with n and f_n as above).

(b) We continue Item (a). In the current interpretation of probability, not only should the relative frequency of an event in a large number of experiments be close to the probability, but there is an obscure secondary stipulation. If the probability of success is very small, we should be practically sure that the event should not occur in a single trial [A.N. Kolmogorov, *Grundbegriffe der Wahrscheinlichkeitsrechnung*, Berlin, 1933; English translation: Chelsea, 1956]. If x is a Bernoulli sequence (result of n independent coin tosses) with a very low relative success frequency f_n/n (the coin is heavily biased), then, almost necessarily, $x_1 = 0$. That is, the assumption that 1 occurs as the very first element implies substantial regularity of the overall sequence.

Show that there is a constant c such that $\delta_0(x) \leq \log n / f_n - c$ implies $x_1 = 0$.

Comments. Hint for Item (b): construct the test that rejects at level 2^{-m} when $x_1 = 1$ and $f_n \leq n2^{-m}$. Show that this is a Bernoulli test. Compare this test with δ_0 in Item (a). For sequential Bernoulli tests for infinite sequences see Exercise 2.5.17. Source: P. Martin-Löf, *Inform. Contr.*, 9(1966), 602–619.

2.5 *Random Infinite Sequences

Consider the question of how C behaves in terms of increasingly long initial segments of a fixed infinite binary sequence (or string) ω. Is it monotone in the sense that $C(\omega_{1:m}) \leq C(\omega_{1:n})$, or $C(\omega_{1:m}|m) \leq C(\omega_{1:n}|n)$, for all infinite binary sequences ω and all $m \leq n$? We have already seen that the answer is negative in both cases. A similar effect arises when we try to use Kolmogorov complexity to solve the problem of finding a proper definition of random infinite sequences (*collectives*) according to the task already set by von Mises in 1919, Section 1.9.

2.5.1 Complexity Oscillations

It is seductive to call an infinite binary sequence ω random if there is a constant c such that for all n, the n-length prefix $\omega_{1:n}$ has $C(\omega_{1:n}) \geq n - c$. However, such sequences do not exist. We shall show that for high-complexity sequences, with $C(\omega_{1:n}) \geq n - \log n - 2 \log\log n$ for all n, this results in so-called *complexity oscillations*, where for every $\epsilon > 0$,

$$\frac{n - C(\omega_{1:n})}{\log n}$$

oscillates between 0 and $1+\epsilon$ for large enough n. First, we show that the C complexity of prefixes of each infinite binary sequence drops infinitely often unboundedly far below its own length.

Theorem 2.5.1 *Let $f : \mathcal{N}^+ \to \mathcal{N}$ be a total recursive function satisfying $\sum_{n=1}^{\infty} 2^{-f(n)} = \infty$ (such as $f(n) = \log n$). Then for all infinite binary sequences ω, we have $C(\omega_{1:n}|n) \leq n - f(n)$ infinitely often.*

Proof. In order to get rid of an $O(1)$ term in the final argument, we first change f into something that gets arbitrarily larger yet still diverges in the same way as f. Define

$$F(n) = \left\lfloor \log \left(\sum_{i=1}^{n} 2^{-f(i)} \right) \right\rfloor .$$

Note that $\sum_{F(n)=m} 2^{-f(n)} \geq 2^m - 1$. Now let g, also total recursive, be defined as $g(n) = f(n) + F(n)$. It follows that

$$\begin{aligned} \sum_{n=1}^{\infty} 2^{-g(n)} &= \sum_{m=1}^{\infty} \sum_{F(n)=m} 2^{-f(n)-m} \\ &\geq \sum_{m=1}^{\infty} 2^{-m}(2^m - 1) = \infty. \end{aligned}$$

Now we come to the real argument. Consider the unit interval $[0,1]$ laid out in a circle so that 0 and 1 are identified. The partial sums $G_n = \sum_{i=1}^{n} 2^{-g(i)}$ mark off successive intervals $I_n \equiv [G_{n-1}, G_n) \bmod 1$ on this circle. We exploit the fact that a point on the circle will be contained in many of these intervals.

For each $x \in \{0,1\}^*$, the associated cylinder is the set

$$\Gamma_x = \{\omega \in \{0,1\}^\infty : \omega_{1:l(x)} = x\}.$$

The geometric interpretation is $\Gamma_x = [0.x, 0.x + 2^{-l(x)})$. Let

$$A_n = \left\{ x \in \{0,1\}^n : \Gamma_x \bigcap I_n \neq \varnothing \right\}.$$

It follows from the divergence of G_n that for any ω there is an infinite set $N \subseteq \mathcal{N}$ consisting of the infinitely many n such that prefixes $\omega_{1:n}$ belong to A_n. Describing a prefix $\omega_{1:n} \in A_n$ by its index in the set, we have

$$\begin{aligned} C(\omega_{1:n}|n) &\leq \log |A_n| + O(1) \leq \log \frac{G_n - G_{n-1}}{2^{-n}} + O(1) \\ &= n - g(n) + O(1) \leq n - f(n). \end{aligned}$$

□

Corollary 2.5.1 Let $f(n)$ be as in Theorem 2.5.1. Then $C(\omega_{1:n}) \leq n - f(n)$ infinitely often, provided $C(n|n - f(n)) = O(1)$ (such as $f(n) = \log n$).

Proof. This is a slightly stronger statement than Theorem 2.5.1. Let p be a description of $\omega_{1:n}$, given n, of length $C(\omega_{1:n}|n)$. Let $f(), g()$ be as in the proof of Theorem 2.5.1, and let q be an $O(1)$-length program that retrieves n from $n - f(n)$. Then $l(\bar{q}p) \leq n - f(n)$, since $l(p) \leq n - g(n)$ and $g(n) - f(n)$ rises unboundedly. We pad $\bar{q}p$ to length $n - f(n)$, obtaining $\bar{q}1^{n-f(n)-l(\bar{q}p)-1}0p$. This is a description for $\omega_{1:n}$. Namely, we first determine $\bar{q}$ to find q. The length of the total description is $n - f(n)$. By assumption, q computes n from this. Given n we can retrieve $\omega_{1:n}$ from p. □

In [P. Martin-Löf, *Z. Wahrsch. Verw. Geb.*, 19(1971), 225–230], Corollary 2.5.1 is stated to hold without the additional condition of n being retrievable from $n - f(n)$ by an $O(1)$-bit program. The proof of this fact is attributed to [P. Martin-Löf, On the oscillation of the complexity of infinite binary sequences (Russian), unpublished, 1965].

There is a simple proof that yields a result only slightly weaker than this. We first prove the result with the particular function $\log n$ substituted for $f(n)$ in the statement of the theorem. We then iterate the construction to prove a version of the theorem with a particular function $g(n)$ substituted for $f(n)$. Our result is almost tight, since for functions $h(n)$ that are only slightly larger than $g(n)$ the sum $\sum 2^{-h(n)}$ is finite. Moreover, the proof is explicit in that we exhibit $g(n)$.

Let ω be an infinite binary sequence, and $\omega_{1:m}$ an m-length prefix of ω. If $\omega_{1:m}$ is the nth binary string in the lexicographic order $0, 1, 00, \ldots$, that is, $n = \omega_{1:m}$, $m = l(n)$, then $C(\omega_{1:n}) \leq C(\omega_{m+1:n}) + c$, with c a constant independent of n and m. Namely, with $O(1)$ additional bits of information, we can trivially reconstruct the nth binary string $\omega_{1:m}$ from the length $n - l(n)$ of $\omega_{m+1:n}$. By Theorem 2.1.2, we find that $C(\omega_{m+1:n}) \leq n - l(n) + c$ for some constant c independent of n, whence the claimed result with $f(n) = \log n$ follows.

It is easy to see that we get a stronger result by iteration of the above argument. There are infinitely many n such that the initial segment $y = \omega_{1:n}$ can be divided as $y = y_1 y_2 \ldots y_k$, where $l(y_1) = 2$, $y_1 = l(y_2)$, $y_2 = l(y_3)$, $\ldots$, $y_{k-1} = l(y_k)$. Use the usual pairing between natural numbers and binary strings. Clearly, given y_k we can easily compute all of $\omega_{1:n}$, by determining y_{k-1} as the binary representation of $l(y_k)$, y_{k-2} as the binary representation of $l(y_{k-1})$, and so on until we obtain $l(y_1) = 2$. (If $\omega_{1:24} = 010011101100000110100001$, then $y_1 = 01$, which corresponds to natural number 4, so $y_2 = 0011$, which corresponds to the natural number 18, and finally $y_3 = 101100000110100001$. Hence, given y_3, we can easily determine all of $\omega_{1:24}$.) Then, for infinitely many n,

$$C(\omega_{1:n}) \leq \kappa + c,$$

for κ determined by $n = \kappa + l(\kappa) + l(l(\kappa)) + \cdots + 2$, all terms greater than or equal to 2. If we put $g(n) = n - \kappa$, then it can be shown that $\sum 2^{-g(n)} = \infty$ but that for only slightly larger functions $h(n) > g(n)$ the sum converges (for example $h(n) = g(n) +$ the number of terms in $g(n)$). There is an interesting connection with prefix codes, Section 1.11.1.

Our approach in this proof makes it easy to say something about the frequency of these complexity oscillations. Define a wave function w by $w(1) = 2$ and $w(i) = 2^{w(i-1)}$. Then the above argument guarantees that there are at least k values $n_1, n_2, \ldots, n_k$ less than $n = w(k)$ such that $C(\omega_{1:n_i}) \leq n_i - g(n_i) + c$ for all $i = 1, 2, \ldots, k$. Obviously, this can be improved.

In Figure 2.4 we display the complexity oscillations of initial segments of high-complexity sequences as they must look according to Theorems 2.5.1, 2.5.4, 2.5.5. The upper bound on the oscillations, $C(\omega_{1:n}) = n + O(1)$, is reached infinitely often for almost every high-complexity

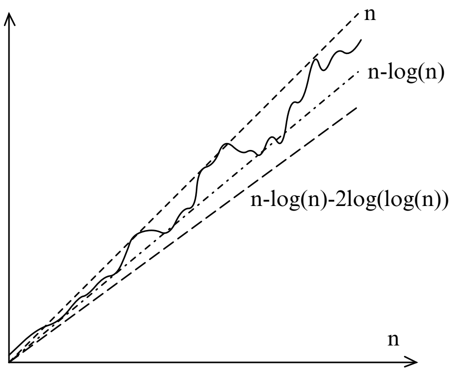

FIGURE 2.4. Complexity oscillations of initial segments of high–complexity infinite sequences

sequence. Furthermore, the oscillations of all high-complexity sequences stay above $n - \log n - 2 \log \log n$, but dip infinitely often below $n - \log n$.

Having shown that the complexity of prefixes of each infinite sequence drops infinitely often unboundedly below the maximum value, we now want to show that Theorem 2.5.1 is optimal. But let's first discuss what this means. Clearly, Theorem 2.5.1 is nontrivial only for very irregular sequences x. It holds trivially for regular sequences such as $\omega = 0^\infty$, where the complexity of the initial segments $\omega_{1:n}$ is about $\log n$. We will prove that it is sharp for those ω that are maximally random. To make this precise, we must define rigorously what we mean by a random infinite sequence. It is of major significance that in so doing we also succeed in completing in a satisfactory way the program outlined by von Mises.

Due to the complexity oscillations, the idea of identifying random infinite sequences with those such that $C(\omega_{1:n}) \geq n-c$, for all n, is trivially infeasible. That is the bad news. In contrast, a similar approach in Section 2.4 for finite binary sequences turned out to work just fine. Its justification was found in Martin-Löf's important insight that to justify any proposed definition of randomness one has 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. Instead of proving each such property separately, one may be able to show, once and for all, that the random sequences introduced possess, in an appropriate sense, all possible properties of stochasticity.

The naive execution of the above ideas in classical mathematics is infeasible as shown by the following example: Consider as sample space S the set of all one-way infinite binary sequences. The cylinder $\Gamma_x = \{\omega : \omega = x \ldots\}$ consists of all infinite binary sequences starting with the finite binary sequence x. For instance, $\Gamma_\epsilon = S$. The uniform distribution λ on the sample space is defined by $\lambda(\Gamma_x) = 2^{-l(x)}$. That is, the probability of an infinite binary sequence ω starting with a finite initial segment x is $2^{-l(x)}$. In probability theory it is general practice that if a certain property, such as the law of large numbers, or the law of the iterated logarithm, has been shown to have probability one, then one calls this a *law of randomness*. For example, in our sample space the law of large numbers says that $\lim_{n\to\infty}(\omega_1 + \cdots + \omega_n)/n = \frac{1}{2}$. If A is the set of elements of S that satisfy the law of large numbers, then it can be shown that $\lambda(A) = 1$.

Generalizing this idea for S with measure μ, one may identify *any* set $B \subseteq S$ such that $\mu(B) = 1$ with a law of randomness, namely, "to be an element of B." Elements of S that do not satisfy the law "to be an element of B" form a set of measure zero, a *null set*. It is natural to call an element of the sample space 'random' if it satisfies all laws of randomness. Now we are in trouble. For each element $\omega \in S$, the set $B_\omega = S - \{\omega\}$ forms a law of randomness. But the intersection of all these sets B_ω of probability one is empty. Thus, no sequence would be random if we require that all laws of randomness that exist be satisfied by a random sequence.

It turns out that a constructive viewpoint enables us to carry out this program mathematically without such pitfalls. In practice, all laws that are proved in probability theory to hold with probability one are effective in the sense of Section 1.7. A straightforward formalization of this viewpoint is to require a law of probability to be partial recursive in the sense that we can effectively test whether it is violated. This suggests that the set of random infinite sequences should not be defined as the intersection of all sets of measure one, but as the intersection of all sets of measure one with a recursively enumerable complement. The latter intersection is again a set of measure one with a recursively enumerable complement. Hence, there is a single effective law of randomness that can be stated as the property "to satisfy all effective laws of randomness," and the infinite sequences have this property with probability one.

2.5.2 Sequential Randomness Tests

As in Section 2.4, we define a test for randomness. However, this time the test will not be defined on the entire sequence (which is impossible for an effective test and an infinite sequence), but for each finite binary string. The value of the test for an infinite sequence is then defined as the maximum of the values of the test on all prefixes. Since this suggests an effective process of sequential approximations, we call it a sequential test. Below, we need to use notions of continuous sample spaces and measures as treated in Section 1.6.

Definition 2.5.1 Let μ be a recursive probability measure on the sample space $\{0,1\}^\infty$. A total function $\delta : \{0,1\}^\infty \to \mathcal{N} \bigcup \{\infty\}$ is a *sequential μ-test* (sequential Martin-Löf μ-test for randomness) if

1. $\delta(\omega) = \sup_{n \in \mathcal{N}}\{\gamma(\omega_{1:n})\}$, where $\gamma : \mathcal{N} \to \mathcal{N}$ is a (total) lower semicomputable function ($V = \{(m, y) : \gamma(y) \geq m\}$ is a recursively enumerable set); and
2. $\mu\{\omega : \delta(\omega) \geq m\} \leq 2^{-m}$, for each $m \geq 0$.

If μ is the uniform measure λ, then we often use simply the term *sequential test.*

We can require γ to be a recursive function without changing the notion of a sequential μ-test. By definition, for each lower semicomputable function γ there exists a recursive function ϕ with $\phi(x, k)$ nondecreasing in k such that $\lim_{k\to\infty} \phi(x, k) = \gamma(x)$. Define a recursive function γ' by $\gamma'(\omega_{1:n}) = \phi(\omega_{1:m}, k)$ with $\langle m, k\rangle = n$. Then, $\sup_{n\in\mathcal{N}}\{\gamma'(\omega_{1:n})\} = \sup_{n\in\mathcal{N}}\{\gamma(\omega_{1:n})\}$.

Example 2.5.1 Consider $\{0,1\}^\infty$ with the uniform measure $\lambda(x) = 2^{-l(x)}$. An example of a sequential λ-test is to test whether there is some 1 in an even position of $\omega \in \{0,1\}^\infty$. Let

$$\gamma(\omega_{1:n}) = \begin{cases} \frac{1}{2}n & \text{if } \sum_{i=1}^{n/2} \omega_{2i} = 0, \\ 0 & \text{otherwise.} \end{cases}$$

The number of x's of length n such that $\gamma(x) \geq m$ is at most $2^{n/2}$ for any $m \geq 1$. Therefore, $\lambda\{\omega : \delta(\omega) \geq m\} \leq 2^{-m}$ for $m > 0$. For $m = 0$, $\lambda\{\omega : \delta(\omega) \geq m\} \leq 2^{-m}$ holds trivially. A sequence ω is random with respect to this test if $\delta(\omega) < \infty$. Thus, a sequence ζ with 0's in all even locations will have $\delta(\zeta) = \infty$ and it will fail the test, and hence ζ is not random with respect to this test. Notice that this is not a very strong test of randomness. For example, a sequence $\eta = 010^\infty$ will pass δ and be considered random with respect to this test. This test filters out only some nonrandom sequences with all 0's at the even locations and cannot detect other kinds of regularities. $\diamond$

We continue the general theory of sequential testing. If $\delta(\omega) = \infty$, then we say that ω *fails* δ, or that δ *rejects* ω. Otherwise, ω *passes* δ. By definition, the set of ω's that are rejected by δ has μ-measure zero, and conversely, the set of ω's that pass δ has μ-measure one.

Suppose that for a test δ we have $\delta(\omega) = m$. Then there is a prefix y of ω, with $l(y)$ minimal, such that $\gamma(y) = m$ for the γ used to define δ. Then obviously, *each* infinite sequence ζ that starts with y has $\delta(\zeta) \geq m$. The set of such ζ is $\Gamma_y = \{\zeta : \zeta = y\rho, \rho \in \{0,1\}^\infty\}$, the cylinder generated by y. Geometrically speaking, Γ_y can be viewed as the set of all real numbers $0.y\ldots$ corresponding to the half-open *interval* $I_y = [0.y, 0.y + 2^{-l(y)})$. For the uniform measure, $\lambda(\Gamma_y)$ is equal to $2^{-l(y)}$, the common length of I_y.

In terms of common statistical tests, the critical regions are formed by the nested sequence $V_1 \supseteq V_2 \supseteq \cdots$, where V_m is defined as $V_m = \{\omega : \delta(\omega) \geq m\}$, for $m \geq 1$. We can formulate the definition of V_m as

$$V_m = \bigcup \{\Gamma_y : (m, y) \in V\}.$$

In geometric terms, V_m is the union of a set of subintervals of $[0, 1)$. Since V is recursively enumerable, so is the set of intervals whose union is V_m. For each critical section we have $\mu(V_m) \leq 2^{-m}$ (in the measure we count overlapping intervals only once).

Now we can reformulate the notion of passing a sequential test δ with associated set V:

$$\delta(\omega) < \infty \text{ iff } \omega \notin \bigcap_{m=1}^{\infty} V_m.$$

Definition 2.5.2 Let $\mathbf{V}$ be the set of all sequential μ-tests. An infinite binary sequence ω, or the binary-represented real number $0.\omega$, is called *μ-random* if it passes *all* sequential μ-tests:

$$\omega \notin \bigcup_{V \in \mathbf{V}} \bigcap_{m=1}^{\infty} V_m.$$

For each sequential μ-test V, we have $\mu(\bigcap_{m=1}^{\infty} V_m) = 0$, by Definition 2.5.1. We call $\bigcap_{m=1}^{\infty} V_m$ a *constructive μ-null set* . Since there are only countably infinitely many sequential μ-tests V, it follows from standard measure theory that

$$\mu\left(\bigcup_{V \in \mathbf{V}} \bigcap_{m=1}^{\infty} V_m\right) = 0,$$

and we call the set $U = \bigcup_{V \in \mathbf{V}} \bigcap_{m=1}^{\infty} V_m$ the *maximal constructive μ-null set.*

In analogy to Section 2.4, we construct a lower semicomputable function $\delta_0(\omega|\mu)$, the universal sequential μ-test that incorporates (majorizes) all sequential μ-tests $\delta_1, \delta_2, \ldots$ and that corresponds to U.

Definition 2.5.3 A *universal sequential μ-test f* is a sequential μ-test such that for each sequential μ-test δ_i there is a constant $c \geq 0$ such that for all $\omega \in \{0,1\}^{\infty}$, we have $f(\omega) \geq \delta_i(\omega) - c$.

Theorem 2.5.2 *There is a universal sequential μ-test* ($denoted\ by\ \delta_0(\cdot|\mu)$).

Proof. Start with the standard enumeration $\phi_1, \phi_2, \ldots$ of partial recursive functions from $\mathcal{N}$ into $\mathcal{N} \times \mathcal{N}$. In this way, we enumerate all recursively enumerable sets of pairs of integers in the form of the ranges of the ϕ_i's. In particular, we have included *any* recursively enumerable set V associated with a sequential μ-test. The *only* thing we have to do is to eliminate those ϕ_i's that do not correspond to a sequential μ-test.

First, effectively modify each ϕ (we drop the subscript for convenience) to a function ψ such that range ϕ equals range ψ, and ψ has the special property that if $\psi(n) < \infty$, then also $\psi(1), \psi(2), \ldots, \psi(n-1) < \infty$.

Second, use each ψ to construct a function $\delta : \{0,1\}^\infty \to \mathcal{N}$ by approximation from below. In the algorithm, at each stage of the computation the arrays γ and δ contain the current approximation to the function values of γ and δ. This is doable because the nonzero part of the approximation is always finite.

Step 1. Initialize γ and δ by setting $\delta(\omega) := \gamma(\omega_{1:n}) := 0$, for all $\omega \in \{0,1\}^\infty, n \in \mathcal{N}$, and set $i := 0$.

Step 2. Set $i := i + 1$; compute $\psi(i)$ and let its value be (y, m).

Step 3. **If** $\gamma(y) \geq m$ **then go to** Step 2 **else** set $\gamma(y) := m$.

Step 4. **If** $\mu(\bigcup_{\gamma(z) \geq k} \Gamma_z) > 2^{-k}$ for some k, $k = 1, \ldots, m$ **{**since μ is a recursive function we can effectively test whether the new assignment violates Definition 2.5.1**}** **then** terminate **{**the computation of δ is finished**}** **else** set $\delta(\omega) := \max\{m, \delta(\omega)\}$, for all $\omega \in \Gamma_y$, and **go to** Step 2.

For the uniform measure $\lambda(\Gamma_x) = 2^{-l(x)}$, the conditional in Step 4 simplifies for $i = 1$ to $m > l(y)$. In case the range of ψ is already a sequential μ-test, then the algorithm never finishes but approximates δ from below. If the range of ψ is not a sequential μ-test, then at some point the conditional in Step 4 is violated and the computation of δ terminates. The resulting δ is a sequential μ-test, even a recursive one. If the conditional in Step 4 is never violated, but the computation of ψ diverges for some argument, then δ is trivially a lower semicomputable sequential μ-test.

Executing this procedure on all functions in the list $\phi_1, \phi_2, \ldots$, we obtain an effective enumeration $\delta_1, \delta_2, \ldots$ of all sequential μ-tests (and only sequential μ-tests). The function $\delta_0(\cdot|\mu)$ defined by

$$\delta_0(\omega|\mu) = \sup_{i \in \mathcal{N}} \{\delta_i(\omega) - i\}$$

is a universal sequential μ-test.

First, $\delta_0(\omega|\mu)$ is a lower semicomputable function, since the set $\{(m, \omega) : \delta_0(\omega|\mu) \geq m\}$ is recursively enumerable. The proof that $\delta_0(\cdot|\mu)$ is a sequential μ-test, and majorizes all other sequential μ-tests additively, is completely analogous to the proof for the similarly defined universal P-test in Section 2.4. □

Any other sequential μ-test δ_i can discover at most a constant additional amount randomness in a sequence ω than does $\delta_0(\cdot|\mu)$. That is, $\delta_i(\omega) \leq \delta_0(\omega|\mu) + i$, for all ω.

The difference between any two universal sequential μ-tests $\delta_0(\cdot|\mu)$ and ${\delta'}_0(\cdot|\mu)$ is bounded by a constant: $|\delta_0(\omega|\mu) - {\delta'}_0(\omega|\mu)| \leq c$, with c independent of ω. We are now ready to separate the random infinite sequences from the nonrandom ones.

Definition 2.5.4 Let the sample space $\{0,1\}^\infty$ be distributed according to μ, and let $\delta_0(\cdot|\mu)$ be a universal sequential μ-test. An infinite binary sequence ω is *μ-random in the sense of Martin-Löf* if $\delta_0(\omega|\mu) < \infty$. We call such a sequence simply *random*, where both μ and Martin-Löf are understood. (This is particularly interesting for μ is the uniform measure.)

Note that *this definition does not depend on the choice of the particular universal sequential μ-test* with respect to which the level is defined. Hence, the line between random and nonrandom infinite sequences is drawn sharply without dependence on a reference μ-test. Clearly, the set of infinite sequences that are not random in the sense of Martin-Löf forms precisely the maximal constructive μ-null set of μ-measure zero we have constructed above. Therefore, we have the following result.

Theorem 2.5.3 *Let μ be a recursive measure. The set of μ-random infinite binary sequences has μ-measure one.*

We say that the universal sequential μ-test $\delta_0(\cdot|\mu)$ rejects an infinite sequence with probability zero, and we conclude that a randomly selected infinite sequence passes all effectively testable laws of randomness with probability one.

The main question remaining is the following: Let λ be the uniform measure. Can we formulate a universal sequential λ-test in terms of complexity? In Theorem 2.4.2 the universal (nonsequential) test is expressed in that way. The most obvious candidate for the universal sequential test would be $f(\omega) = \sup_{n \in \mathcal{N}}\{n - C(\omega_{1:n})\}$, but it is improper. To see this, it is simplest to notice that $f(\omega)$ would declare *all* infinite ω to be nonrandom, since $f(\omega) = \infty$, for all ω, by Theorem 2.5.1. The same would be the case for $f(\omega) = \sup_{n \in \mathcal{N}}\{n - C(\omega_{1:n}|n)\}$, by about the same proof. It is difficult to express a universal sequential test precisely in terms of C-complexity. But in Chapter 3 we show that it is easy to separate the random infinite sequences from the nonrandom ones in terms of prefix complexity.

2.5.3 Characterization of Random Sequences

How accurately can we characterize the set of infinite random sequences in complexity terms? It turns out that we can sandwich them between a proper superset in Theorem 2.5.4, page 152, and a proper subset in Theorem 2.5.5, page 153. First we bound the amplitude of the oscillations of random sequences.

Definition 2.5.5 The infinite series $\sum 2^{-f(n)}$ is *recursively convergent* if there is a recursive sequence $n_1, n_2, \ldots$ such that

$$\sum_{n=n_m}^{\infty} 2^{-f(n)} \leq 2^{-m}, \ m = 1, 2, \ldots .$$

Theorem 2.5.4 *Let $f(n)$ be a recursive function such that $\sum_{n=1}^{\infty} 2^{-f(n)} < \infty$ is recursively convergent. If an infinite binary sequence ω is random with respect to the uniform measure, then $C(\omega_{1:n}|n) \geq n - f(n)$, from some n onward.*

Proof. We define a sequential test that is passed only by the ω's satisfying the conditions in the theorem. For each m, let the critical section V_m consist of all infinite binary sequences ω such that there exists an $n \geq n_m$ for which $C(\omega_{1:n}|n) < n - f(n)$. In other words, V_m consists of the union of all intervals $[0.\omega_{1:n}, 0.\omega_{1:n} + 2^{-n})$ satisfying these conditions. We have to show that this is a sequential test. We can recursively enumerate the intervals that constitute V_m, and therefore $V_1, V_2, \ldots$ is a sequence of recursively enumerable sets. Obviously, the sequence is nested. For every large enough n, at most $2^{n-f(n)}$ strings y of length n satisfy $C(y|n) < n - f(n)$ (Theorem 2.2.1). Hence, with λ the uniform measure, we have, for all m,

$$\lambda(V_m) \leq \sum_{n=n_m}^{\infty} 2^{n-f(n)} 2^{-n} \leq 2^{-m}.$$

Therefore, the sequence of critical regions forms a sequential test, and $\lambda(\bigcap_{m=1}^{\infty} V_m) = 0$. That is, $\bigcap_{m=1}^{\infty} V_m$ is a constructive λ-null set associated with a sequential test. Consequently, it is contained in the maximal constructive λ-null set, which consists precisely of the sequences that are not random according to Martin-Löf. $\square$

We can say fairly precisely which functions f satisfy the condition in Theorem 2.5.4. Examples are $f(n) = 2 \log n$ and $f(n) = \log n + 2 \log \log n$. In fact, the function $g(n)$ used in the proof of Theorem 2.5.1 is about the borderline. It is *almost* sufficient that $f(n) - g(n)$ be unbounded for a recursive function $f(n)$ to satisfy the condition in Theorem 2.5.4. A precise form of the borderline function is given in Exercise 3.6.7 on page 231.

With Theorem 2.5.4 we have shown that Theorem 2.5.1 is optimal in the sense that it gives the deepest complexity dips to be expected from sequences that are random with respect to the uniform measure (in the sense of Martin-Löf). But also, we have found a property of infinite sequences in terms of C that is implied by randomness with respect to the uniform measure. Is there also a property of infinite sequences in terms of complexity C that implies randomness with respect to the uniform measure?

Theorem 2.5.5 *Let ω be an infinite binary sequence.*

(i) *If there exists a constant c such that $C(\omega_{1:n}) \geq n - c$, for infinitely many n, then ω is random in the sense of Martin-Löf with respect to the uniform measure.*

(ii) *The set of ω for which there exists a constant c and infinitely many n such that $C(\omega_{1:n}) \geq n - c$ has uniform measure one.*

Proof. We first prove the following claim:

Claim 2.5.1 Let $\omega \in \{0,1\}^\infty$. There exists a positive constant c such that $C(\omega_{1:n}|n) \geq n - c$ for infinitely many n iff there exists a positive constant c such that $C(\omega_{1:n}) \geq n - c$ for infinitely many n.

Proof. (Only if) This is the easy direction, since conditional information does not increase complexity. Hence, for all ω, n, we have $C(\omega_{1:n}|n) \leq C(\omega_{1:n})$ up to a fixed additive constant.

(If) For some fixed constant c_1, we have for all ω, n that $C(\omega_{1:n}) \leq C(\omega_{1:n}|n) + 2l(n - C(\omega_{1:n}|n)) + c_1$. (The right-hand side of the inequality is the length of a description of $\omega_{1:n}$.) Since in the 'If' direction we assume that $C(\omega_{1:n}) \geq n - c$ for some c and infinitely many n, we obtain $n - C(\omega_{1:n}|n) \leq c + c_1 + 2l(n - C(\omega_{1:n}|n))$ for this infinite sequence of n's. But that is possible only if there is a constant c_2 such that $n - C(\omega_{1:n}|n) \leq c_2$ for the same infinite sequence of n's, which finishes the proof. □

(i) Below, a 'test' is a 'λ-test' with λ the uniform measure. We denote sequential tests by δ's, and (nonsequential) tests of Section 2.4 by γ's. Let $\delta_0(\cdot|\lambda)$ denote the universal sequential test with respect to the uniform measure λ, and let $\gamma_0(\cdot|L)$ denote the universal test with respect to the uniform distribution L.

Since a sequential test is a fortiori a test, there is a constant c such that $\delta_0(\omega_{1:n}|\lambda) \leq \gamma_0(\omega_{1:n}|L) + c$, for all ω and n. By choosing the specific universal test of Theorem 2.4.2, we have $\delta_0(\omega_{1:n}|\lambda) \leq n - C(\omega_{1:n}|n)$ up to a constant. Since δ_0 is monotonic nondecreasing,

$$\lim_{n \to \infty} \delta_0(\omega_{1:n}|\lambda) \leq \liminf_{n \to \infty} (n - C(\omega_{1:n}|n)) + O(1).$$

For those ω's satisfying the assumption in the statement of the theorem, by Claim 2.5.1 the right-hand side of the inequality is finite. By Theorem 2.4.2, therefore, such ω's are random with respect to the uniform measure.

(ii) For each c and n, let $V_{c,n}$ denote the union of the set of intervals associated with prefixes $\omega_{1:n}$ of infinite binary sequences ω such that $C(\omega_{1:n}|n) \geq n - c$. Let λ be the uniform measure. There are at most 2^{n-c} strings of length less than $n - c$ and therefore at least $2^n - 2^{n-c}$ strings x of length n satisfying $C(x|n) \geq n - c$. Hence, for each m and c we have

$$\lambda\left(\bigcup_{n=m}^{\infty} V_{c,n}\right) \geq \lambda\left(V_{c,m}\right) \geq (2^m - 2^{m-c})2^{-m} = 1 - 2^{-c}.$$

Since the right-hand term is independent of m, we also have

$$\lambda\left(\bigcap_{m=1}^{\infty} \bigcup_{n=m}^{\infty} V_{c,n}\right) \geq 1 - 2^{-c}.$$

Since

$$\bigcap_{m=1}^{\infty} \bigcup_{n=m}^{\infty} V_{c,n} \subseteq \bigcap_{m=1}^{\infty} \bigcup_{n=m}^{\infty} V_{c+1,n}$$

for all positive integers c, we obtain

$$\begin{aligned}\lambda\left(\bigcup_{c=1}^{\infty} \bigcap_{m=1}^{\infty} \bigcup_{n=m}^{\infty} V_{c,n}\right) &= \lim_{c\to\infty} \lambda\left(\bigcap_{m=1}^{\infty} \bigcup_{n=m}^{\infty} V_{c,n}\right) \\ &\geq \lim_{c\to\infty} (1 - 2^{-c}) = 1.\end{aligned}$$

The formula $\bigcup_{c=1}^{\infty} \bigcap_{m=1}^{\infty} \bigcup_{n=m}^{\infty} V_{c,n}$ denotes precisely the set of infinite sequences ω for which there exists a positive integer constant c such that for infinitely many n, $C(\omega_{1:n}|n) \geq n - c$ holds. A fortiori, the same holds without the conditional up to an additive constant. □

We conclude that the set of sequences satisfying the condition in Theorem 2.5.4 contains the set of sequences that are random with respect to the uniform measure (in Martin-Löf's sense), and the latter contains the set of sequences satisfying the condition in Theorem 2.5.5; see Figure 2.5. There, the inner oval is the set of sequences satisfying Theorem 2.5.5; the middle oval is the set of Martin-Löf random sequences; the outer oval is the set of sequences satisfying Theorem 2.5.4. Containment is always proper. The outer oval in its turn is properly contained in the set defined by Exercise 2.5.5 on page 159. Although the differences between

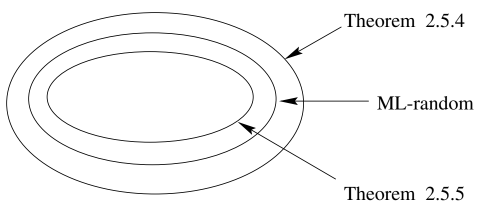

FIGURE 2.5. Three notions of 'chaotic' infinite sequences

each pair of the three sets are nonempty, they are not large, since all three sets have uniform measure one. For instance, the set of random sequences not satisfying the condition of Theorem 2.5.5 has uniform measure zero. In Example 3.6.19 on page 237 it is shown that the condition involved precisely characterizes a stronger notion of randomness than Martin-Löf randomness.

The combination of Theorems 2.5.4 and 2.5.5 enables us to give a relation between the upward and downward oscillations of the complexity of prefixes of the random sequences satisfying the property in Theorem 2.5.5 as follows:

Corollary 2.5.2 If f is a recursive function such that $\sum 2^{-f(n)}$ converges recursively and $C(\omega_{1:n}) \geq n - c$ for some constant c and for infinitely many n, then $C(\omega_{1:n}) \geq n - f(n)$ from some n onward.

The universal sequential μ-test characterizes the set of infinite random sequences. There are other ways to do so. We give an explicit characterization of infinite random sequences with respect to the uniform measure in Theorem 3.6.1 and its corollary, page 222. This characterization is an exact expression in terms of the prefix complexity developed in Chapter 3.

Apart from sequential tests as developed above there are other types of tests for randomness of individual infinite sequences. The extended theory of randomness tests can be given only after we have treated lower semicomputable semimeasures in Sections 4.5.7 and 4.5.6. There we give exact expressions for μ-tests for randomness, for arbitrary recursive μ.

We recall von Mises's classic approach to obtaining infinite random sequences ω as treated in Section 1.9, which formed a primary inspiration to the work reported in this section. It is of great interest whether one can, in his type of formulation, capture the intuitively and mathematically satisfying notion of infinite random sequence in the sense of Martin-Löf. According to von Mises, an infinite binary sequence ω is random (a collective) if

1. ω has the property of frequency stability with limit p; that is, if $f_n = \omega_1 + \omega_2 + \cdots + \omega_n$, then the limit of f_n/n exists and equals p.

2. Any subsequence of ω chosen according to an admissible place-selection rule has frequency stability with the same limit p as in condition 1.

One major problem was how to define 'admissible,' and one choice was to identify it with Church's notion of selecting a subsequence $\zeta_1\zeta_2\ldots$ of $\omega_1\omega_2\ldots$ by a partial recursive function ϕ by $\zeta_n = \omega_m$ if $\phi(\omega_{1:r}) = 0$ for precisely $n-1$ instances of r with $r < m-1$ and $\phi(\omega_{1:m-1}) = 0$. We called these ϕ 'place-selection rules according to von Mises–Wald–Church,' and the resulting sequences ζ Mises–Wald–Church random.

Definition 2.5.6 *Mises–Wald–Church stochastic sequences* are Mises–Wald–Church random sequences with limiting frequency $\frac{1}{2}$.

In Section 1.9 we stated that there are Mises–Wald–Church stochastic sequences that do not satisfy effectively testable properties of randomness such as the law of the iterated logarithm or the infinite recurrence property. (Such properties are by definition satisfied by sequences that are Martin-Löf random.) In fact, the distinction between the Mises–Wald–Church stochastic sequences and the Martin-Löf random ones is quite large, since there are Mises–Wald–Church stochastic sequences ω such that $C(\omega_{1:n}) = O(f(n)\log n)$ for every unbounded, nondecreasing, total recursive function f; see also Exercise 2.5.13 on page 161. Such Mises–Wald–Church stochastic sequences are very nonrandom sequences from the viewpoint of Martin-Löf randomness, where one requires that $C(\omega_{1:n})$ be asymptotic to n. See R.P. Daley, *Math. Systems Theory*, 9(1975), 83–94. Note, that although a Mises–Wald–Church stochastic sequence may have very low Kolmogorov complexity, it in fact has very high time-bounded Kolmogorov complexity. See Exercise 7.1.7 on page 546.

If we consider also sequences with limiting frequencies different from $\frac{1}{2}$, then it is obvious that there are sequences that are random according to Mises–Wald–Church, but not according to Martin-Löf. Namely, *any* sequence ω with limiting relative frequency p has complexity $C(\omega_{1:n}) \leq H(p)n + o(n)$, where $H(p) = p\log 1/p + (1-p)\log 1/(1-p)$ ($H(p)$ is Shannon's binary entropy). This means that for each $\epsilon > 0$ there are Mises–Wald–Church random sequences ω with $C(\omega_{1:n}) < \epsilon n$ for all but finitely many n.

On the other hand, clearly all Martin-Löf random sequences are also Mises–Wald–Church stochastic (each admissible selection rule is an effective sequential test).

This suggests that we have to liberate our notion of admissible selection rule somewhat in order to capture the proper notion of an infinite random sequence using von Mises's approach. A proposal in this direction was given by A.N. Kolmogorov [*Sankhyā*, Ser. A, 25(1963), 369–376] and D.W. Loveland [*Z. Math. Logik Grundl. Math.* 12(1966), 279–294].

Definition 2.5.7 A *Kolmogorov–Loveland admissible selection function* to select an infinite subsequence $\zeta_1\zeta_2\ldots$ from $\omega = \omega_1\omega_2\ldots$ is a partial recursive function $\phi : \{0,1\}^* \to \mathcal{N} \times \{0,1\}$ from binary strings to (index, bit) pairs (not necessarily defined on all of $\{0,1\}^*$). The subsequence selection is a two-phase process. First we select an intermediate sequence z of elements of ω. Initially, $z = \epsilon$. If $z = z_1z_2\ldots z_m$ is the intermediate sequence selected after m steps, and $\phi(z) = (i,a)$ ($a \in \{0,1\}$), then $z\omega_i$ is the intermediate sequence selected after $m+1$ steps. However, ϕ is partial, so $\phi(z)$ may not be defined. Moreover, we are not allowed to select the same bit position more than once. If for some z either $\phi(z)$ is undefined, or $\phi(z) = (i, \cdot)$ while $\phi(z') = (i, \cdot)$ for some proper initial segment z' of z, then the process terminates with finite z; otherwise z is infinite. If $z = \omega_{i_1}\omega_{i_2}\ldots$, selected by the sequence of associated ϕ-values $\phi(\epsilon) = (i_1, a_1)$, $\phi(\omega_{i_1}) = (i_2, a_2)$, $\ldots$, then we obtain the target selected subsequence ζ by erasing every ω_{i_j} in z with associated $a_j = 0$. The resulting ζ may be finite or infinite, but it is only the infinite ζ in which we are interested. If ω is such that for every Kolmogorov–Loveland admissible selection function the selected sequence $\zeta_1\zeta_2\ldots$—if infinite—has the same limiting frequency as the original ω, and we assume the limiting frequency $\frac{1}{2}$, then ω is called a *Kolmogorov–Loveland stochastic sequence.*

The term 'Kolmogorov–Loveland random sequence' is currently used for infinite binary sequences for which there is no computable nonmonotonic betting strategy that succeeds on it. The strategy has success if it obtains unbounded gain in the limit while betting successively on the nonmonotonically selected bits of the sequence. This is in contrast to the above 'stochastic' definition, which expresses the weaker requirement that there be no computable nonmonotonic selection rule that selects an infinite biased sequence from the original sequence. For more details, see [W. Merkle, J.S. Miller, A. Nies, J. Reimann, F. Stephan, *Ann. Pure Appl. Logic*, 138(2006), 183–210].

As compared to the Mises–Wald–Church approach, the liberation of the selection rule mechanism is contained in the fact that the order of succession of the terms in the subsequence chosen is not necessarily the same as that of the original sequence. Thus, the Kolmogorov–Loveland selection rules are called *nonmonotonic.* In comparison, it is not obvious whether a subsequence $\zeta_1\zeta_2\ldots$ selected from a Kolmogorov–Loveland stochastic sequence $\omega_1\omega_2\ldots$ by a Kolmogorov–Loveland place-selection rule is itself a Kolmogorov–Loveland stochastic sequence. Note that the analogous property necessarily holds for Mises–Wald–Church stochastic sequences. This matter was resolved in [W. Merkle, *J. Symbol. Logic*, 68(2003), 1362–1376], where it was shown that there is a Kolmogorov–Loveland stochastic sequence from which one can select effectively (and in fact monotonically) a subsequence that is no longer Kolmogorov–Loveland stochastic.

Clearly, the set of Kolmogorov–Loveland stochastic sequences is contained in the set of Mises–Wald–Church stochastic sequences. In turn, the set of Kolmogorov–Loveland stochastic sequences contains the set of Martin-Löf random sequences. If $\omega_1\omega_2\ldots$ is Kolmogorov–Loveland stochastic, then clearly $\zeta_1\zeta_2\ldots.$ defined by $\zeta_i = \omega_{\sigma(i)}$, with σ being a recursive permutation, is also Kolmogorov–Loveland stochastic. The Mises–Wald–Church notion of stochasticity does not have this important property of randomness of staying invariant under recursive permutation. Loveland gave the required counterexample in the cited reference. Hence, the containment of the set of Kolmogorov–Loveland stochastic sequences in the set of Mises–Wald–Church stochastic sequences is proper. This follows also from the cited result that the Kolmogorov–Loveland stochastic sequences are not closed under monotonic effective selection rules, earlier observed by A.K. Shen; see the acknowledgments in [W. Merkle, *Ibid.*].

This leaves the question whether the containment of the set of Martin-Löf random sequences in the set of Kolmogorov–Loveland stochastic sequences is proper. Kolmogorov has stated in [*Problems Inform. Transmission*, 5(1969), 3–4] without proof that there exists a Kolmogorov–Loveland stochastic sequence ω such that $C(\omega_{1:n}) = O(\log n)$. But An.A. Muchnik (1958–2007)—not to be confused with his father A.A. Muchnik—showed that this is false, since no ω with $C(\omega_{1:n}) \leq cn + O(1)$ for a constant $c < 1$ can be Kolmogorov–Loveland stochastic. Nonetheless, containment is proper, since A.K. Shen [*Soviet Math. Dokl.*, 38:2(1989), 316–319] has shown that there exists a Kolmogorov–Loveland stochastic sequence that is not random in Martin-Löf's sense. Therefore, the problem of giving a satisfactory definition of infinite Martin-Löf random sequences in the form proposed by von Mises has not yet been solved. See also [A.N. Kolmogorov and V.A. Uspensky, *Theory Probab. Appl.*, 32(1987), 389–412; V.A. Uspensky, A.L. Semenov, and A.K. Shen, *Russ. Math. Surveys*, 45:1(1990), 121–189; An.A. Muchnik, A.L. Semenov, V.A. Uspensky, *Theoret. Comput. Sci.*, 2:207(1998), 1362–1376].

Exercises

2.5.1. [13] Consider $\{0,1\}^\infty$ under the uniform measure. Let $\omega = \omega_1\omega_2 \ldots \in \{0,1\}^\infty$ be random in the sense of Martin-Löf.

(a) Show that $\zeta = \omega_n\omega_{n+1} \ldots$ is Martin-Löf random for each n.

(b) Show that $\zeta = x\omega$ is Martin-Löf random for each finite string x.

Comments. Source: C. Calude, I. Chitescu, *Bolletino U.M.I.*, (7) 3-B(1989), 229–240.

2.5.2. [21] Consider $\{0,1\}^\infty$ under the uniform measure. Let $\omega = \omega_1\omega_2 \ldots \in \{0,1\}^\infty$.

(a) Show that if there is an infinite recursive set I such that either for all $i \in I$ we have $\omega_i = 0$ or for all $i \in I$ we have $\omega_i = 1$, then ω is not random in the sense of Martin-Löf.

(b) Show that if the set $\{i : \omega_i = 0\}$ contains an infinite recursively enumerable subset, then ω is not random in the sense of Martin-Löf.

Comments. Source: C. Calude and I. Chitescu, *Ibid.*

2.5.3. [21] Let $\omega = \omega_1\omega_2 \ldots$ be any infinite binary sequence. Define $\zeta = \zeta_1\zeta_2 \ldots$ by $\zeta_i = \omega_i + \omega_{i+1}$, $i \geq 1$. This gives a sequence over the alphabet $\{0,1,2\}$. Show that ζ is not random in the sense of Martin-Löf under the uniform measure (extend the definition from binary to ternary sequences).

Comments. Hint: the blocks 02 and 20 do not occur in ζ. Source: R. von Mises, *Probability, Statistics and Truth*, Dover, 1981.

2.5.4. [23] Let ω be any infinite binary sequence. Show that for all constants c there are infinitely many m such that for all n with $m \leq n \leq 2m$, $C(\omega_{1:n}) \leq n - c$.

Comments. We are guaranteed to find long complexity oscillations (of length m) in an infinite binary sequence ω relatively near the beginning (namely $\omega_{m:2m}$), even if ω is Martin-Löf random. Source: H.P. Katseff and M. Sipser, *Theoret. Comput. Sci.*, 15(1981), 291–309.

2.5.5. [M19] Let f be such that $\sum 2^{-f(n)} < \infty$. Show that the set of infinite binary sequences ω satisfying $C(\omega_{1:n}|n) \geq n - f(n)$ for all but finitely many n has uniform measure 1.

Comments. Hint: The number of y with $l(y) = n$ such that $C(y) < n - f(n)$ is less than $2^{n-f(n)}$. This implies that the probability that this inequality is satisfied is less than $2^{-f(n)}$, and the result follows by the Borel–Cantelli lemmas; see Exercise 1.10.2 on page 64. This set of ω's properly contains the set defined by Theorem 2.5.4. Source: P. Martin-Löf, *Z. Wahrsch. Verw. Geb.*, 19(1971), 225–230.

2.5.6. [09] Consider infinite binary sequences ω with respect to the uniform measure. Show that with probability one there exists a constant c such that $C(\omega_{1:n}|n) \geq n - c$ for infinitely many n.

Comments. Hint: use Theorem 2.5.5, Item (ii), and Claim 2.5.1. Source: P. Martin-Löf, *Z. Wahrsch. Verw. Geb.*, 19(1971), 225–230. .

2.5.7. [19] Consider infinite binary sequences ω with respect to the uniform measure. Show that if f is a recursive function and $\sum 2^{-f(n)}$ converges recursively and $C(\omega_{1:n}) \geq n - c$ for some constant c and infinitely many n, then $C(\omega_{1:n}) \geq n - f(n)$ for all but finitely many n.

Comments. This formulation establishes a connection between upward and downward oscillations of the complexity of prefixes of almost all (random) infinite binary sequences. For f we can take $f(n) = 2 \log n$, or $f(n) = \log n + 2 \log \log n$, and so on. Hint: combine Theorems 2.5.4, 2.5.5. Source: P. Martin-Löf, *Z. Wahrsch. Verw. Geb.*, 19(1971), 225–230.

2.5.8. [19] Show that there exists an infinite binary sequence ω and a constant $c > 0$ such that $\liminf_{n\to\infty} C(\omega_{1:n}|n) \leq c$, but for any unbounded function f we have $\limsup_{n\to\infty} C(\omega_{1:n}|n) \geq n - f(n)$.

Comments. The oscillations can have amplitude $\Omega(n)$. Hint: use the n-strings defined above. Construct $\omega = y_1 y_2 \ldots$ from finite strings y_i. Let c be a fixed independent constant. For odd i choose y_i such that $y_1 \ldots y_i$ is an n-string, which implies that $C(y_{1:i}|l(y_{1:i})) < c$. For even i choose y_i as a long enough random string so that $C(y_{1:i}|l(y_{1:n})) \geq l(y_{1:n}) - f(l(y_{1:n}))$. Source: H.P. Katseff and M. Sipser, *Theoret. Comput. Sci.*, 15(1981), 291–309.

2.5.9. [39] Consider the Lebesgue measure λ on the set of intervals contained in $[0, 1)$ defined by $\lambda(\Gamma_y) = 2^{-l(y)}$. (Recall that for each finite binary string y the cylinder Γ_y is the set of all infinite strings ω starting with y.) Let ω be an infinite binary sequence such that for every recursively enumerable sequence $A_1, A_2, \ldots$ of sets of intervals with the property that the series $\sum_i \lambda(A_i) < \infty$ converges, ω is contained in only finitely many A_i. Show that for the ω's defined this way, the *Solovay random sequences* are precisely the infinite binary sequences that are random in the sense of Martin-Löf with respect to the uniform measure.

Comments. In Martin-Löf's definition of randomness (with respect to the uniform measure) of infinite binary sequences, he required that $\lambda(A_i) \leq 2^{-i}$. That definition is equivalent to stipulating the existence of some regulator of convergence $f(i) \to \infty$ that is recursive and nondecreasing such that $\lambda(A_i) \leq 2^{-f(i)}$. Solovay's definition has the advantage that it does not require such a regulator. Source: R.M. Solovay, *Lecture Notes*, 1975, unpublished; and G.J. Chaitin, *Algorithmic Information Theory*, Cambridge University Press, 1987; A. K. Shen, *Soviet Math. Dokl.*, 38:2(1989), 316–319.

2.5.10. [35] (a) Show that for every positive constant c there is a positive constant c' such that $\{\omega_{1:n} : C(\omega_{1:n}; n) \geq n - c\} \subseteq \{\omega_{1:n} : C(\omega_{1:n}|n) \geq n - c'\}$.

(b) Use the observation in Item (a) to show that Theorem 2.5.4 holds for the uniform complexity measure $C(\cdot; l(\cdot))$.

(c) Show that if f is a recursive function and $\sum 2^{-f(n)} = \infty$, then for all infinite ω we have $C(\omega_{1:n}; n) \leq n - f(n)$ for infinitely many n. Hence, Theorem 2.5.1 holds for uniform complexity.

Comments. Hint for Item (a): define the notion of a (universal) uniform test as a special case of Martin-Löf's (universal) test. Compare this result with the other exercises to conclude that whereas the uniform complexity tends to be higher in the low-complexity region, the length-conditional complexity tends to be higher in the high-complexity region. Source: D.W. Loveland, *Inform. Contr.*, 15(1969), 510–526.

2.5.11. • [31] Show that the following statements are equivalent for an infinite binary sequence ω: For some constant c and infinitely many n, possibly different in each statement,

$$C(\omega_{1:n}|n) \geq n - c,$$
$$C(\omega_{1:n}; n) \geq n - c,$$
$$C(\omega_{1:n}) \geq n - c.$$

Comments. Hint: use Claim 2.5.1 and Exercise 2.5.10. In view of Theorem 2.5.5 these conditions equivalently imply that ω is random in the sense of Martin-Löf. Source: R.P. Daley, pp. 113–122 in: *Computational Complexity*, ed. R. Rustin, *Courant Comput. Sci. Symp.* 7(1971).

2.5.12. [30] (a) Show that the following statements are equivalent for an infinite binary sequence ω: There exists a c such that for infinitely many n, possibly different in each statement,

$$C(\omega_{1:n}|n) \leq c,$$
$$C(\omega_{1:n}; n) \leq l(n) + c,$$
$$C(\omega_{1:n}) \leq l(n) + c.$$

The sequences thus defined are called *pararecursive sequences.*

(b) Show that no pararecursive sequence is random in the sense of Martin-Löf.

Comments. Comparison with the other exercises shows that the recursive sequences are contained in the pararecursive sequences. It also shows that the pararecursive sequences have the cardinality of the continuum, so this containment is proper. R.P. Daley [*J. Symb. Logic,* 41(1976), 626–638] has shown that the lower semicomputable sequences sequences (which properly include the characteristc sequences of recursively enumerable sets) are pararecursive, and this containment is proper by the same argument as before. Hint for Item (b): use Theorem 2.5.4. Despite Item (b), there are pararecursive sequences that are close to being random. For any unbounded function f, there is a pararecursive sequence ω such that for infinitely many n we have $C(\omega_{1:n}|n) \geq n - f(n)$; see Exercise 2.5.8 on page 159. Source: H.P. Katseff and M. Sipser, *Theoret. Comput. Sci.*, 15(1981), 291–309.

2.5.13. [43] Let A be the set of Mises–Wald–Church stochastic sequences (with $p = \frac{1}{2}$). The admissible place-selection rules are the partial recursive functions.

(a) Show that there is an $\omega \in A$ such that for each unbounded, nondecreasing, total recursive function f, we have $C(\omega_{1:n}; n) \leq f(n) \log n$ from some n onward.

(b) Show that for all $\omega \in A$, there is a constant c such that $C(\omega_{1:n}; n) \geq \log n - c$ from some n onward.

Consider the larger class $B \supset A$ that is defined just like A but with the admissible place-selection rules restricted to the total recursive functions.

(c) Show that there is an $\omega \in B$ such that $C(\omega_{1:n}; n) \leq f(n)$ from some n onward, for each f as in Item (a).

(d) Show that for each $\omega \in B$, for each constant c, we have $C(\omega_{1:n}; n) \geq c$ from some n onward.

Comments. Compare this with the text following Definition 2.5.6 on page 156. This shows that there are Mises–Wald–Church stochastic sequences of quite low complexity, and that it makes a difference whether the admissible place-selection rules are partial recursive or total recursive. Source: R.P. Daley, *Math. Systems Theory*, 9 (1975), 83–94. Item (a) is proved using Item (c), which is attributed to D.W. Loveland, and uses a construction (LMS algorithm) in D.W. Loveland, *Z. Math. Logik*, 12(1966), 279–294. Compare with Exercise 7.1.7, page 546, to see what happens when we impose a total recursive time bound on the decoding process.

2.5.14. [35] Show that there is no Mises–Wald–Church stochastic sequence ω (with limiting frequency $\frac{1}{2}$) and with $C(\omega_{1:n}) = O(\log n)$.

Comments. This exercise was open in the second edition of this book, solved in [W. Merkle, *J. Comput. Syst. Sci.*, 74:3(2008), 350–357]. Compare Exercise 2.5.13.

2.5.15. [33] (a) Show that there exists an infinite binary sequence ω that is random with respect to the uniform measure, but for each constant c there are only finitely many n such that $C(\omega_{1:n}|n) > n - c$ (the condition of Theorem 2.5.5 does not hold).

(b) Show that there exists an infinite binary sequence ω satisfying (i) $C(\omega_{1:n}) > n - f(n)$ from some n onward and $\sum 2^{-f(n)}$ converges recursively (the condition in Theorem 2.5.4 holds), and (ii) ω is not random with respect to the uniform measure.

Comments. Thus, each containment in the nested sequence of sets of infinite binary sequences that satisfy the condition in Theorem 2.5.4, randomness according to Martin-Löf, and the condition in Theorem 2.5.5, as in Figure 2.5, is proper. Source, C.P. Schnorr, *Math. Systems Theory*, 5(1971), 246–258.

2.5.16. [21] (a) Show that none of the variants of algorithmic complexity, such as $C(x)$, $C(x|l(x))$, and $C(x;l(x))$, is invariant with respect to cyclic shifts of the strings.

(b) Show that all these variants coincide to within the logarithm of the minimum of all these measures.

Comments. Hint: use the idea in the proof of Theorem 2.5.4. This invariance cannot be expected from any complexity measure in this book at all. Source: C.P. Schnorr, pp. 193–211 in: R.E. Butts and J. Hintikka, eds., *Basic Problems in Methodology and Linguistics*, D. Reidel, 1977.

2.5.17. [36] (a) Consider an infinite sequence of zeros and ones generated by independent tosses of a coin with probability p $(0 < p < 1)$ for 1. Define sequential Bernoulli tests (in analogy with Section 2.5 and Exercise 2.4.4 on page 142). Show that there exists a universal sequential Bernoulli test δ_0. An infinite binary sequence ω is a Bernoulli sequence if $\delta_0(\omega) < \infty$. Show that the set of Bernoulli sequences has measure one with respect to the measure induced in the set of infinite binary sequences interpreted as independent $(p, 1-p)$ Bernoulli trials.

(b) In our definition of infinite Bernoulli sequences no restrictions were laid on the limiting behavior of the relative frequency, such as, for instance, required in Condition 1 of Definition 1.9.1 of an infinite random sequence (collective) ω, where we require that $\lim_{n\to\infty} f_n/n = p$ for some p $(0 < p < 1)$ (Section 1.9). The relative frequency f_n/n of ones in increasingly longer prefixes $\omega_{1:n}$ of a collective ω does not oscillate indefinitely, but converges to a definite limit. Show that remarkably, this is also the case for an infinite Bernoulli sequence ω.

(c) By the law of large numbers, *all* real numbers p in $[0, 1]$ occur as limit frequencies $\lim_{n\to\infty} f_n/n$ for infinite random binary sequences ω, and not only the recursive ones. Show that in contrast, for infinite Bernoulli sequences ω, the limit relative frequency cannot vanish, $\lim_{n\to\infty} f_n/n = 0$, unless $\omega_n = 0$ for all n.

Comments. Hint for Item (b): for an arbitrary rational $\epsilon > 0$ construct a sequential Bernoulli test that rejects at level 2^{-m} if $|f_i/i - f_j/j| > \epsilon$, for some $i, j \geq h(m)$, for some suitable nondecreasing total recursive function. Compare with the universal Bernoulli test of Exercise 2.4.4 on page 142. Hint for Item (c): this is the infinite analogue of the phenomenon in Item (b) of Exercise 2.4.4 on page 142. From Item (c) we conclude that an event with vanishing limit frequency is actually impossible. This is in stark contrast with von Mises's explicit statement of the opposite for his conception of random sequences (collectives) [R. von Mises, *Probability, Statistics and Truth*, Dover, 1981 (Reprinted)]. Source: P. Martin-Löf, *Inform. Contr.*, 9(1966), 602–619. Additionally we mention the following result [L.A. Levin, *Sov. Math. Dokl.* , 14(1973), 1413–1416]. Suppose we are given a constructively closed family $\mathcal{M}$ of measures (this notion is defined naturally on the space of measures). Let a test f be called *uniform* for $\mathcal{M}$ if for all measures in $\mathcal{M}$, for all positive integers k, the measure of outcomes ω where $f(\omega) > k$ is at most 2^{-k}. There exists a universal uniform test.

2.5.18. [37] Let μ be a recursive measure on the sample space $\{0, 1\}^\infty$. Recall from Section 2.5 Martin-Löf's construction of a constructive μ-null set using a notion of sequential test V with associated critical regions $V_1 \supseteq V_2 \supseteq \cdots$ of measures $\mu(V_i) \leq 2^{-i}$, for $i \geq 1$. A constructive μ-null set is a *total recursive μ-null set* if additionally, $f(i) = \mu(V_i)$ is a total recursive function. Call an infinite sequence *μ-random in the sense of Schnorr* if it is not contained in any total recursive μ-null set.

(a) Show that there is no universal total recursive μ-null set that contains all others.

(b) Show that the set of sequences that are μ-random in the sense of Martin-Löf is a subset of the set of sequences that are μ-random in the sense of Schnorr.

(c) An $\omega \in \{0,1\}^\infty$ is an *atom* with respect to μ if $\mu(\omega) > 0$. A measure μ is called *discrete* if the set of atoms of μ has μ-measure one. (Obviously all atoms of recursive μ are recursive sequences.) Show that the Schnorr-μ-random sequences coincide with the Martin-Löf-μ-random sequences iff μ is discrete.

Comments. Item (c) implies that for continuous μ (such as the uniform measure), Schnorr randomness is weaker than Martin-Löf randomness. The notion of total recursive null sets is the recursive analogue of the intuitionistic notion of sets of measure zero by L.E.J. Brouwer [A. Heyting, *Intuitionism, an Introduction*, North-Holland, 1956]. Sometimes the following statement is called *Schnorr's thesis:* "A sequence behaves within all effective procedures like a μ-random sequence iff it is μ-random in the sense of Schnorr." Source: C.P. Schnorr, pp. 193–211 in: R.E. Butts and J. Hintikka, eds., *Basic Problems in Methodology and Linguistics*, D. Reidel, 1977.

2.5.19. [M42] We abstract away from levels of significance and concentrate on the arithmetic structure of statistical tests. Statistical tests are just Π_n^0 null sets, for some n (Exercise 1.7.21, page 46). The corresponding definition of randomness is defined as, "an infinite binary sequence is Π_n^0-random with respect to a recursive measure μ if it is not contained in any Π_n^0 set V with μ-measure zero." Has the set of Π_n^0-random sequences μ-measure one?

Comments. Source: H. Gaifman and M. Snir, *J. Symb. Logic*, 47(1982), 495–548.

2.5.20. [M43] We assume familiarity with the unexplained notions below. We leave the arithmetic hierarchy of Exercise 2.5.19 and consider hyperarithmetic sets. Define an infinite binary sequence to be *hyperarithmetically random* if it belongs to the intersection of all hyperarithmetic sets of measure one. (A hyperarithmetic set can be regarded as a constructive version of the restriction to Borel sets that is usually accepted in probability theory—Section 1.6. The specific Borel sets considered there are always obtained by applying the Borelian operations to recursive sequences of previously defined sets, which means precisely that they are hyperarithmetical.)

(a) Show that the set of sequences that are hyperarithmetically random is a Σ_1^1 set of measure one (Σ_1^1 in the *analytic hierarchy*).

(b) Show that a hyperarithmetic sequence is not hyperarithmetically random.

(c) Show that the set of hyperarithmetically random sequences is not hyperarithmetical.

Comments. Already A. Wald proposed to sharpen von Mises's notion of randomness by defining a sequence to be random if it possesses all properties of probability one that are expressible within a certain formalized logic such as that of *Principia Mathematica*. This exercise is a variation on this idea. Just as here, Wald's proposal can be expected to define a set of random strings that is no longer expressible in the language with which we started. (This does not happen for Martin-Löf random sequences as defined in Section 2.5 because of the existence of a universal sequential test.) However, with the present proposal, the resulting class of random strings, while escaping the hyperarithmetic hierarchy, does not escape us completely but belongs to a class of sets that can still be handled constructively. Source: P. Martin-Löf, pp. 73–78 in: *Intuitionism and Proof Theory*, A. Kino et al., eds., North-Holland, 1970. For related work more in the direction of Wald's ideas, see [P.A. Benioff, *J. Math. Phys.*, 17:5(1976), 618–628, 629–640; L. Longpré and V. Kreinovich, "Randomness as incompressibility: a non-algorithmic analogue," Tech. Rept. UTEP-CS-96-19, Univ. Texas El Paso, 1996]

2.6 Statistical Properties of Finite Sequences

Each individual infinite sequence generated by a $(\frac{1}{2}, \frac{1}{2})$ Bernoulli process (flipping a fair coin) has (with probability 1) the property that the relative frequency of zeros in an initial n-length segment goes to $\frac{1}{2}$ as n goes to infinity. Such randomness-related statistical properties of individual (high)-complexity finite binary sequences are often required in applications of incompressibility arguments. The situation for infinite random sequences is better studied, and therefore we look there first.

Definition 2.6.1 E. Borel has called an infinite sequence of zeros and ones *normal* in the scale of two if for each k, the frequency of occurrences of each block y of length k in the initial segment of length n goes to limit 2^{-k} as n grows unboundedly.

It is known that normality is not sufficient for randomness, since Champernowne's sequence 123456789101112... is normal in the scale of ten. On the other hand, it is universally agreed that a random infinite sequence must be normal. (If not, then some blocks occur more frequently than others, which can be used to obtain better than fair odds for prediction.)

We know from Section 2.5 that each infinite sequence that is random with respect to the uniform measure satisfies all effectively testable properties of randomness: it is normal, it satisfies the so-called law of the iterated logarithm, the number of 1's minus the number of 0's in an initial n-length segment is positive for infinitely many n and negative for another infinitely many n, and so on. While the statistical properties of infinite sequences are simple corollaries of the theory of Martin-Löf randomness, for finite sequences the situation is less simple. Here, we determine the frequencies of occurrence of substrings in strings of high Kolmogorov complexity. In Section 6.4.1 the similar question is treated for subgraphs of high-Kolmogorov-complexity graphs.

2.6.1 Statistics of 0's and 1's

In the finite case, randomness is a matter of degree, because it would be clearly unreasonable to say that a sequence x of length n is random and to say that a sequence y obtained by flipping the first bit 1 in x is nonrandom. What we can do is to express the degree of incompressibility of a finite sequence in the form of its Kolmogorov complexity, and then analyze the statistical properties of the sequence—for example, the number of 0's and 1's in it.

Since almost all finite sequences have about maximal Kolmogorov complexity, each individual maximal-complexity sequence must possess approximately the expected (average) statistical properties of the overall set. For example, we can a priori state that each high-complexity finite binary sequence is normal in the sense that each binary block of length k occurs about equally frequently for k relatively small. In particular, this holds for $k = 1$. However, in many applications we need to know exactly what 'about' and the 'relatively small' in this statement mean. In other words, we are interested in the extent to which Borel normality holds in relation to the complexity of a finite sequence.

Let x have length n. By Example 2.4.4, if $C(x|n) = n + O(1)$, then the number of zeros it contains is

$$\frac{n}{2} + O(\sqrt{n}).$$

Notation 2.6.1 The quantity $K(x|y)$ in this section satisfies

$$C(x|y) \leq K(x|y) \leq C(x|y) + 2\log C(x|y) + 1.$$

We can think of it as roughly the length of a self-delimiting version of a program p of length $l(p) = C(x|y)$. In Chapter 3 it is defined as 'prefix complexity.'

Definition 2.6.2 The class of *deficiency* functions is the set of functions $\delta : \mathcal{N} \rightarrow \mathcal{N}$ satisfying $K(n, \delta(n)|n - \delta(n)) \leq c_1$ for all n. (Hence, $C(n, \delta(n)|n - \delta(n)) \leq c_1$ for all n.)

In this way, we can retrieve n and $\delta(n)$ from $n - \delta(n)$ by a self-delimiting program of at most c_1 bits. We choose c_1 so large that each monotone sublinear recursive function that we are interested in, such as $\log n$, $\sqrt{n}$, $\log\log n$, is such a deficiency function. The constant c_1 is a benchmark that stays fixed throughout this section.

We denote the number of 1's in a binary string $x \in \{0,1\}^*$ by $\#\text{ones}(x)$.

Lemma 2.6.1 *There is a constant c such that for all deficiency functions δ, for each n and $x \in \{0,1\}^n$, if $C(x) \geq n - \delta(n)$, then*

$$\left|\#\text{ones}(x) - \frac{n}{2}\right| \leq \sqrt{\frac{3}{2}(\delta(n)+c)n/\log e}. \tag{2.3}$$

Proof. A general estimate of the tail probability of the binomial distribution, with s_n the number of successful outcomes in n experiments with probability of success $0 < p < 1$, is given by Chernoff's bounds, Lemma 1.10.1 on page 61:

$$\Pr(|s_n - pn| > m) \leq 2e^{-m^2/3pn}. \tag{2.4}$$

Let s_n be the number of 1's in the outcome of n fair coin flips, which means that $p = \frac{1}{2}$. Define $A = \{x \in \{0,1\}^n : |\#\text{ones}(x) - \frac{1}{2}n| > m\}$ and apply Equation 2.4 to obtain

$$d(A) \leq 2^{n+1}e^{-2m^2/3n}.$$

We choose m such that for some constant c to be determined later,

$$\frac{2m^2 \log e}{3n} = \delta(n) + c.$$

We can compress any $x \in A$ in the following way:

1. Let s be a self-delimiting program to retrieve n and $\delta(n)$ from $n - \delta(n)$, of length at most c_1.
2. Given n and $\delta(n)$, we can effectively enumerate A. Let i be the index of x in such an effective enumeration of A. The length of the (not necessarily self-delimiting) description of i satisfies

$$\begin{aligned} l(i) \leq \log d(A) &\leq n + 1 - (2m^2 \log e)/3n \\ &= n + 1 - \delta(n) - c. \end{aligned}$$

The string si is padded to length $n + 1 - \delta(n) - c + c_1$. From si we can reconstruct x by first using $l(si)$ to compute $n - \delta(n)$, then computing n and $\delta(n)$ from s and $n - \delta(n)$, and subsequently enumerating A to obtain the ith element. Let T be the Turing machine embodying the procedure for reconstructing x. Then by Theorem 2.1.1,

$$C(x) \leq C_T(x) + c_T \leq n + 1 - \delta(n) - c + c_1 + c_T.$$

Choosing $c = c_1 + c_T + 2$, we obtain $C(x) < n - \delta(n)$, which contradicts the condition of the theorem. Hence, $|\#\text{ones}(x) - \frac{1}{2}n| \le m$. □

It may be surprising at first glance, but there are no maximally complex sequences with about an equal number of zeros and ones. Equal numbers of zeros and ones is a form of regularity, and therefore lack of complexity. That is, for $x \in \{0,1\}^n$, if $|\#\text{ones}(x) - \frac{1}{2}n| = O(1)$, then the randomness deficiency $\delta(n) = n - C(x)$ is nonconstant (order $\log n$).

Lemma 2.6.2 *There is a constant c such that for all n and all $x \in \{0,1\}^n$, if*

$$\left|\#\text{ones}(x) - \frac{n}{2}\right| \le 2^{-\delta(n)-c}\sqrt{n},$$

then $C(x) \le n - \delta(n)$.

Proof. Let $m = 2^{-\delta(n)-c}\sqrt{n}$, with c a constant to be determined later. Let $A = \{x \in \{0,1\}^n : |\#\text{ones}(x) - \frac{1}{2}n| \le m\}$. There is a constant c_2 such that there are only

$$d(A) \le (2m+1)\binom{n}{n/2} \le c_2 \frac{2^n m}{\sqrt{n}} \tag{2.5}$$

elements in A (use Stirling's approximation, Exercise 1.5.4 on page 17). Thus, for each $x \in A$, we can encode x by its index in an enumeration of A. We can find A from n and $\delta(n)$. We can find n and $\delta(n)$ from $n - \delta(n)$ by a self-delimiting program of size at most c_1. Altogether, this description takes $\log d(A) + c_1 = n - \delta(n) - c + c_1 + \log c_2$ bits. Let this process of reconstructing x be executed by Turing machine T. Choosing $c = c_1 + \log c_2 + c_T$ we obtain by Theorem 2.1.1,

$$C(x) \le C_T(x) + c_T \le n - \delta(n).$$

□

Example 2.6.1 We consider some particular values of $\delta(n)$. Set $\delta_1(n) = \frac{1}{2}\log n - \log\log n$. If $|\#\text{ones}(x) - \frac{1}{2}n| = O(\log n)$, then $C(x) \le n - \delta_1(n) + O(1)$. Set $\delta_2(n) = \frac{1}{2}\log n$. If

$$\left|\#\text{ones}(x) - \frac{n}{2}\right| = O(1),$$

then $C(x) \le n - \delta_2(n) + O(1)$. That is, if the number of 1's is too close to the number of 0's, then the complexity of the string drops significantly below its maximum. ◇

An incompressible string of length n cannot have precisely or almost $\frac{1}{2}n$ ones by Lemma 2.6.2. Then how many ones should an incompressible string contain? The next lemma shows that for an incompressible x having $j + \frac{1}{2}n$ ones, $K(j|n)$ must be at least about order $\log n$.

Lemma 2.6.3 *There is a constant c such that for all n and all $x \in \{0,1\}^n$, if*

$$\left|\#\text{ones}(x) - \frac{n}{2}\right| = j,$$

then $C(x|n) \leq n - \frac{1}{2}\log n + K(j|n) + c$.

Proof. Let $A = \{x \in \{0,1\}^n : |\#\text{ones}(x) - \frac{1}{2}n| = j\}$. There is a constant c_3 such that there are

$$d(A) \leq \binom{n}{n/2} \leq c_3 \frac{2^n}{\sqrt{n}} \tag{2.6}$$

elements in A (use Stirling's approximation, Exercise 1.5.4 on page 17). In order to enumerate elements in A, we need only to describe j and n. Thus, for any $x \in A$, we can encode x by its index i (in $\log d(A)$ bits) in an enumeration of A. With n given, we can recover x from an encoding of j in $K(j|n)$ bits, followed by i. This description of x, given n, takes $\log d(A) + K(j|n) \leq n - \frac{1}{2}\log n + \log c_3 + K(j|n)$ bits. Let T be the Turing machine embodying this procedure to recover x given n. Choosing $c = \log c_3 + c_T$, we have

$$C(x|n) \leq C_T(x|n) + c_T \leq n - \frac{1}{2}\log n + K(j|n) + c.$$

□

Example 2.6.2 For $j = O(1)$ we have $C(x|n) \leq n - \frac{1}{2}\log n + O(1)$, which is slightly stronger than the statement about unconditional $C(x)$ in Example 2.6.1. For $j = O(\sqrt{n})$ and j incompressible ($K(j|n) \geq \frac{1}{2}\log n$), we have $C(x|n) \leq n - O(1)$. Only for such j's is it possible that a number x is incompressible. ◇

2.6.2 Statistics of Blocks

The analysis up till now has been about the statistics of 0's and 1's. But in a normal infinite binary sequence, according to Definition 2.6.2 on page 166, each block of length k occurs with limiting frequency 2^{-k}. That is, blocks 00, 01, 10, and 11 should occur about equally often, and so on. Finite sequences will generally not be exactly normal, but normality will be a matter of degree. We investigate the block statistics for finite binary sequences.

Definition 2.6.3 Let $x = x_1 \ldots x_n$ be a binary string of length n, and y a much smaller string of length l. Let $p = 2^{-l}$ and $\#y(x)$ be the number of (possibly overlapping) distinct occurrences of y in x. For convenience, we assume that x wraps around, so that an occurrence of y starting at the end of x and continuing at the start also counts.

Theorem 2.6.1 *Assume the notation of Definition 2.6.3 with $l \leq \log n$. There is a constant c such that for all n and $x \in \{0,1\}^n$, if $C(x) \geq n - \delta(n)$, then*

$$|\#y(x) - pn| \leq \sqrt{\alpha p n},$$

with $\alpha = [K(y|n) + \log l + \delta(n) + c]3l/\log e$.

Proof. We prove by contradiction. Assume that n is divisible by l. (If it is not, then we can put x on a Procrustean bed to make its length divisible by l at the cost of having the above frequency estimate $\#y(x)$ plus or minus an error term of at most $l/2$.) There are l ways of dividing (the ring) x into $N = n/l$ contiguous equal-sized blocks, each of length l. For each such division $i \in \{0, 1, \ldots, l-1\}$, let $\#y(x,i)$ be the number of (now nonoverlapping) occurrences of block y. We apply the Chernoff bound, Equation 2.4, again. With $A = \{x \in \{0,1\}^n : |\#y(x,i) - pN| > m\}$ this gives $d(A) \leq 2^{n+1}e^{-m^2/3pN}$. We choose m such that for some constant c to be determined later,

$$\frac{m^2 \log e}{3pN} = K(\langle y, i\rangle|n) + \delta(n) + c.$$

To describe an element x in A, we now need only to enumerate A and indicate the index of x in such an enumeration. The description contains the following items:

1. *A description used to enumerate A.* Given $n - \delta(n)$, we can retrieve n and $\delta(n)$ using a self-delimiting description of at most c_1 bits. To enumerate A, we also need to know i and y. Therefore, given $n - \delta(n)$, the required number of bits to enumerate A is at most

$$K(\langle y, i, \delta(n), n\rangle | n - \delta(n)) \leq K(\langle y, i\rangle|n) + c_1.$$

2. *A description of the index of x.* The number of bits to code the index j of x in A is

$$\begin{aligned}\log d(A) &\leq \log\left(2^{n+1}e^{-m^2/3pN}\right)\\ &= n + 1 - \frac{m^2 \log e}{3pN}\\ &= n + 1 - K(\langle y, i\rangle|n) - \delta(n) - c.\end{aligned}$$

This total description takes at most $n + 1 - \delta(n) - c + c_1$ bits. Let T be a Turing machine reconstructing x from these items. According to Theorem 2.1.1, therefore

$$C(x) \leq C_T(x) + c_T \leq n + 1 - \delta(n) - c + c_1 + c_T.$$

With $c = c_1 + c_T + 2$ we have $C(x) < n - \delta(n)$, which contradicts the assumption of the theorem. Therefore, $|\#y(x,i) - pN| \le m$, which in turn implies

$$|\#y(x,i) - pN| \le \sqrt{\frac{K(\langle y,i\rangle|n) + \delta(n) + c}{\log e} 3pN}.$$

For each division i ($0 \le i \le l-1$) this inequality follows from the $\delta(n)$ incompressibility of x. Notwithstanding the fact that occurrences of substrings in different divisions are dependent, the inequality holds for each division separately and independently. The theorem now follows by noting that $|\#y(x) - pn| = \sum_{i=0}^{l-1} |\#y(x,i) - pN|$, $K(\langle y,i\rangle|n) \le K(y|n) + K(i|n) + O(1)$ and $K(i|n) \le \log l + O(1)$. □

Similar to the analysis of blocks of length 1, the complexity of a string drops below its maximum in case some block y of length l occurs in one of the l block divisions, say i, with frequency exactly pN ($p = 1/2^l$). Then we can point out x by giving n, y, i, and its index in a set of cardinality

$$\binom{N}{pN}(2^l - 1)^{N-pN} = O\left(\frac{2^{Nl}}{\sqrt{p(1-p)N}}\right).$$

Therefore,

$$C(x|\langle n,y\rangle) \le n - \frac{1}{2}\log n + \frac{1}{2}(l + 3\log l) + O(1).$$

2.6.3 Length of Runs

It is known from probability theory that in a randomly generated finite sequence the *expectation* of the length of the longest run of zeros or ones is pretty high. For each individual finite sequence with high Kolmogorov complexity we are *certain* that it contains each block (say, a run of zeros) up to a certain length.

Theorem 2.6.2 *Let x of length n satisfy $C(x) \ge n - \delta(n)$. Then for sufficiently large n,* each *block y of length*

$$l = \log n - \log\log n - \log(\delta(n) + \log n) - O(1)$$

occurs at least once in x.

Proof. We are sure that y occurs at least once in x if $\sqrt{\alpha pn}$ in Theorem 2.6.1 is less than pn. This is the case if $\alpha < pn$, that is,

$$\frac{K(y|n) + \log l + \delta(n) + O(1)}{\log e} 3l < pn.$$

$K(y|n)$ is majorized by $l + 2\log l + O(1)$ (since $K(y|n) \leq K(y) + O(1)$) and $p = 2^{-l}$ with l set at

$$l = \log n - \log(3\delta(n)\log n + 3\log^2 n)$$

(which equals l in the statement of the theorem up to an additive constant). Substitution yields

$$\frac{l + 3\log l + \delta(n) + O(1)}{\log e} 3l < 3(\delta(n)\log n + \log^2 n),$$

and it is easy to see that this holds for sufficiently large n. □

Corollary 2.6.1 If $\delta(n) = O(\log n)$, then *each* block of length $\log n - 2\log\log n - O(1)$ occurs at least once in x.

In Lemma 6.9.1 we show that if $C(x|n,p) \geq n$ then no substring of length greater than $2\log n$ occurs (possibly overlapping) twice in x. Here, $n = l(x)$, and p is some fixed program used to reconstruct x from a description of length $C(x|n,p)$ and n.

Analyzing the proof of Theorem 2.6.2, we can improve the corollary for low values of $K(y|n)$.

Corollary 2.6.2 If $\delta(n) = O(\log\log n)$, then for each $\epsilon > 0$ and every large enough n, every string x of length n contains an all-zero run y (for which $K(y|n) = O(\log l)$) of length $l = \log n - (1+\epsilon)\log\log n + O(1)$.

Since there are $2^n(1 - O(1/\log n))$ strings x of length n with $C(x) \geq n - \log\log n + O(1)$, the *expected length of the longest run of consecutive zeros* if we flip a fair coin n times is at least l as in Corollary 2.6.2.

We show in what sense Theorem 2.6.2 is sharp. Let $x = uvw$, $l(x) = n$, and $C(x) \geq n - \delta(n)$. We can describe x by giving

1. A description of v in $K(v)$ bits;
2. The literal representation of uw;
3. A description of $l(u)$ in $\log n + \log\log n + 2\log\log\log n + O(1)$ bits.

Then, since we can find n by $n = l(v) + l(uw)$,

$$\begin{aligned} C(x) \leq n - l(v) + K(v) + \log n \\ + (1 + o(1))\log\log n + O(1). \end{aligned} \tag{2.7}$$

Substitute $C(x) = n - \delta(n)$ and $K(v) = o(\log\log n)$ (choose v to be very regular) in Equation 2.7 to obtain

$$l(v) \leq \delta(n) + \log n + (1 + o(1)) \log\log n.$$

This means that for instance, for each $\epsilon > 0$, no maximally complex string x with $C(x) = n + O(1)$ contains a run of zeros (or the initial binary digits of π) of length $\log n + (1+\epsilon)\log\log n$ for n large enough and regular enough. By Corollary 2.6.2, on the other hand, such a string x *must* contain a run of zeros of length $\log n - (1+\epsilon)\log\log n + O(1)$.

Exercises

2.6.1. [24] The great majority of binary strings of length n have a number of 0's in between $\frac{1}{2}n - \sqrt{n}$ and $\frac{1}{2}n + \sqrt{n}$. Show that there are x's of length n with $\frac{1}{2}n + \Omega(\sqrt{n})$ 0's, with $C(x) = n + O(1)$.

2.6.2. [29] Let $\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} \omega_i = p$ for an infinite binary sequence $\omega = \omega_1\omega_2\ldots$, for some p between 0 and 1 (compare Section 1.9).

(a) Show that if $C(\omega_{1:n}) \sim n$, then $p = \frac{1}{2}$.

(b) Show that if $p = \frac{1}{4}$, then $C(\omega_{1:n}) \leq 0.82n$ asymptotically.

(c) Show that in general, if $c = p\log 1/p + (1-p)\log 1/(1-p)$, then $C(\omega_{1:n}) \leq cn + o(n)$. If p is about $\frac{1}{4}$, then $C(\omega_{1:n}) \leq 0.80n + o(n)$.

Comments. Source: P. Gács, *Lecture Notes on Descriptional Complexity and Randomness*, Manuscript, Boston University, 1987; attributed to A.N. Kolmogorov. Hint: Item (a), use Lemma 2.6.2.

2.6.3. [M35] A finite binary string x of length n is called *δ-random* if $C(x|n) \geq n - \delta$. A Turing machine place-selection rule R is a Turing machine that selects and outputs a (not necessarily consecutive) substring $R(x)$ from its input x. If R is the kth Turing machine in the standard enumeration, then $C(R) = C(k)$.

Show that for any $\epsilon > 0$, there exist numbers n_0 and $\mu > 0$ such that if $l(x) = n$, $l(R(x)) = r \geq n_0$, $\sum_{1\leq i\leq r} R(x)_i = m$, and $(\delta + C(R|n))/r < \mu$, then

$$\left|\frac{m}{r} - \frac{1}{2}\right| < \left(\frac{\delta + C(R|n) + 2.5\log r}{(2\log e - \epsilon)r}\right)^{1/2}.$$

Comments. δ-random sequences were introduced by A.N. Kolmogorov, *Lect. Notes Math.*, Vol. 1021, Springer-Verlag, 1983, 1–5. He noted that "sequences satisfying this condition have for sufficiently small δ the particular property of frequency stability in passing to subsequences." In this exercise we supply some quantitative estimates of frequency stability. Source: E.A. Asarin, *SIAM Theory Probab. Appl.*, 32(1987), 507–508. This exercise is used by Asarin to show that δ-random elements of certain finite sets obey familiar probability-theoretic distribution laws. In particular, for some particular finite sets each δ-random element is ϵ-normal; see also E.A. Asarin, *Soviet Math. Dokl.*, 36(1988), 109–112.

2.7 Algorithmic Properties of C

By algorithmic properties of C we mean properties with a recursion-theoretic flavor as in Section 1.7. We have already met a few of these. By Theorem 2.3.1, the greatest monotonic lower bound on $C(n)$ is unbounded, but goes to infinity more slowly than any monotonic unbounded partial recursive function. By Theorem 2.3.2 the integer function $C(n)$ is not recursive. Nonetheless, by Theorem 2.3.3 the integer function $C(n)$ can be approximated arbitrarily closely from above; it is upper semicomputable. Unfortunately, at each stage of such an approximation process, for each size of error, there are infinitely many x such that the approximation of $C(x)$ and its real value are at least this error apart.

In fact, a much stronger statement holds: For each total recursive function f with $\lim_{x\to\infty} f(x) = \infty$ the set of x for which we can prove $C(x) > f(x)$ is finite (Theorem 2.7.1 Item (iii) below). Thus, if we choose $f(x) \ll l(x)$, then we know that $C(x) \gg f(x)$ for almost all x of each length, Theorem 2.2.1, yet we can prove this only for finitely many x.

Theorem 2.7.1 (i) *The set $A = \{(x,a) : C(x) \leq a\}$ is recursively enumerable, but not recursive.*

(ii) *Every partial recursive function $\phi(x)$ that is a lower bound on $C(x)$ is bounded.*

(iii) *Let $f(x)$ be a total recursive function with $g(x) \leq f(x) \leq l(x)$ for all x and some unbounded monotonic function g. Then the set $B = \{x : C(x) \leq f(x)\}$ is simple. That is, B is recursively enumerable and the complement of B is infinite but does not contain an infinite recursively enumerable subset.*

Proof. (i) That A is recursively enumerable follows immediately from Theorem 2.3.3. However, A is not recursive. Namely, if A is recursive, then we can compute $C(x)$ by asking the consecutive questions "is $C(x) \leq a$?" for $a := 0, 1, \ldots$, contradicting Theorem 2.3.2.

(ii) Let ϕ be a partial recursive function and define $D = \{x : \phi(x) \leq C(x)\}$. If D is finite, there is nothing to prove. Assume that D is infinite and ϕ is unbounded, by way of contradiction. Recursively enumerate the domain of definition of ϕ without repetition, and define a total recursive function g by $g(n)$ that equals the least x in this enumeration such that $\phi(x) \geq n$. For each n there is such an x, by the contradictory assumption.

If $\phi = \phi_k$ in the standard effective enumeration $\phi_1, \phi_2, \ldots$ of the partial recursive functions, as in Section 1.7, then $n \leq C(x) \leq l(n) + l(\bar{k})$, up to a constant. For n large enough we have a contradiction.

(iii) That B is recursively enumerable follows from Item (i). The complement of B is infinite by Theorem 2.2.1. We prove that B is simple. Let D be an infinite recursively enumerable set contained in the complement of B. The restriction $f_D(x)$ of $f(x)$ to D is a partial recursive lower bound for $C(x)$. By Item (ii), therefore $f_D(x)$ is bounded. Since $f(x)$ rises unboundedly with x this is possible only if D is finite. □

Corollary 2.7.1 The set RAND defined by $\{x : C(x) \geq l(x)\}$ is immune—it is infinite and has no infinite recursively enumerable subset. In fact, the proofs support stronger results in that the set B above is *effectively* simple and RAND is *effectively* immune (Exercise 2.7.6).

2.7.1 Undecidability by Incompressibility

This approach allows us to give a result similar to Lemma 1.7.6 on page 35, but with examples of undecidable statements that differ from the ones given by Gödel. Namely, for each formal system T, there is a constant c_T such that no formula of form "$C(x) \geq c_T$" is provable in T.

Example 2.7.1 If T is an axiomatizable sound theory 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 that demonstrably possesses this property. Formally,

- Let T be an axiomatizable theory (T is a recursively enumerable set consisting of axioms and provable formulas). Hence, there is a k such that T is describable in k bits: $C(T) \leq k$.
- Let T be sound: all formulas in T are true (in the standard model of the natural numbers).
- Let $S_c(x)$ be a formula in T with the meaning "x is the lexicographically least binary string of length c with $C(x) \geq c$." Here x is a formal parameter and c an explicit constant, so $C(S_c) \leq \log c$ up to a fixed constant independent of T and c.

For each c, there exists an x such that $S_c(x) = \textbf{true}$ is a true statement by a simple counting argument (Theorem 2.3.1). 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, for each candidate string y of length c, we can decide $S_c(y) = \textbf{true}$ (which is the case for $y = x$) or $\neg S_c(y) = \textbf{true}$ (which is the case for $y \neq x$) by simple enumeration of all proofs in T. (Here we use the soundness of T.) We need to distinguish the descriptions of T and S_c. We can do this by coding T's description in self-delimiting format; see Equation 1.4. This takes not more than $2k$ bits. Hence, for some fixed constant c' independent of T and c, we obtain $C(x) \leq 2k + \log c + c'$, which contradicts $C(x) > c$ for all $c > c_T$, where $c_T = 3k + c'$ for another constant c'. (A minor improvement of the argument shows that $c_T = k + 2\log k + c'$ suffices.) ◇

Corollary 2.7.2 There is a recursively enumerable set B with an infinite complement such that for every axiomatizable sound theory T there are only finitely many n for which the formula "$n \notin B$" is both true and provable in T. (But with finitely many exceptions, all infinitely many such formulas are true.)

Proof. Let B be the simple set in Theorem 2.7.1, Item (iii), and let $\bar{B}$ be its complement. Clearly, the set $D \subseteq \bar{B}$ of elements n that can be proved in T to belong to $\bar{B}$ is recursively enumerable. Since B is simple, its complement $\bar{B}$ does not contain an infinite recursively enumerable subset. Therefore, D is finite, which proves the theorem. □

We have formulated Corollary 2.7.2 so as to bring out some similarities and differences with Lemma 1.7.6 as clearly as possible. As pointed out in Section 1.7, the set K_0 used in Lemma 1.7.6 is complete, whereas the set B used in Corollary 2.7.2 is simple. According to generally accepted viewpoints in recursion theory, the set K_0 is different from the set B in an essential way. Therefore, we can regard the proofs of the existence of undecidable statements in sufficiently rich axiomatizable theories by Lemma 1.7.6 and Corollary 2.7.2 as essentially different.

The set K_0 is not only complete in the sense of Turing reducibility, it is also complete in the sense of many-to-one reducibility. A set A is *many-to-one* reducible to a set B if there exists a recursive function f such that for all x, $x \in A$ iff $f(x) \in B$. A set A is complete in the sense of many-to-one reducibility, *m-complete* for short, if A is recursively enumerable and all recursively enumerable sets B are many-to-one reducible to A. As it turns out, m-complete sets are nonrecursive. This raises the question, are all nonrecursive recursively enumerable sets m-complete? The answer was given by E. Post in 1944 by introducing simple sets as the first examples of nonrecursive recursively enumerable sets that are not m-complete. (For the notions of reducibility, completeness, and simple sets, see the exercises in Section 1.7, in particular Exercises 1.7.16 and 1.7.15.)

The set B used in Corollary 2.7.2 is a simple set and hence not m-complete. Therefore, although B is many-to-one reducible to K_0, the set K_0 is not many-to-one reducible to B. This shows that K_0 is of a so-called higher degree of unsolvability with respect to many-to-one reducibility than B, which is the substance of the viewpoint that they differ in an essential way.

In less formal terms the approaches are different because the first one can be viewed as a form of *Russell's paradox* and the other one as a form of the *Richard–Berry paradox*. Both paradoxes are described in [B. Russell and A.N. Whitehead, *Principia Mathematica*, Oxford, 1917]. While the first paradox formed the original incentive for the authors to supply the sophisticated logical foundation for set theory in the *Principia*, in a footnote they state that the second paradox "was suggested to us by Mr. G.G. Berry of the Bodleian Library."

The paradox due to Bertrand Russell (1872–1970) arises when the collection of all sets that are not members of themselves is considered as a set. If this collection is a member of itself, then it contradicts the set definition, but if it is not a member of itself, then by the set definition it is a member of itself (which is a contradiction as well). There is a close connection between Russell's paradox and the result of Gödel cited as Lemma 1.7.6 on page 35. We have seen that this result was proved by reducing the halting problem in the form of K_0 to the decision problem in a sufficiently strong, sound, axiomatized theory. Since K_0 is m-complete, this shows that any problem shown to be unsolvable *in this way* must have a degree of unsolvability at least as high as the maximal degree of unsolvability with respect to many-to-one reducibility as any recursively enumerable set.

The Richard–Berry paradox is the definition of a number as "the least number that cannot be defined in fewer than twenty words." Formalizing the notion of 'definition' as the shortest program from which a number can be computed by the reference machine U, it turns out that the quoted statement (reformulated appropriately) is not an *effective description*. This was essentially what we did in the proof of Corollary 2.7.2, by reducing the set B to the decision problem in a sufficiently strong, sound, axiomatizable theory. But B is of a lesser degree of unsolvability with respect to many-to-one reducibility than is K_0. Therefore, showing undecidability of sufficiently rich axiomatizable theories using Kolmogorov complexity *in this way* is essentially different from Gödel's original approach.

Gödel's first incompleteness theorem entails an *explicit* construction of a statement s, associated with each sufficiently strong, sound, axiomatized theory T, that is undecidable in T. Formula s simply says of itself "I am unprovable in T." In contrast, the construction in Corollary 2.7.2 says that for any sound axiomatized system T there is a constant $c_T < \infty$ such that *all* true statements with the meaning "$C(x) \geq c_T$" are unprovable in T. By Theorem 2.3.1 there are infinitely many such statements. Now suppose we have an effective procedure to find such constants for given theories, that is, a total recursive function ϕ such that $\phi(T) \geq c_T$ for all T. Then, unfortunately, Theorem 2.7.1, Item (ii), tells us that *no* effective procedure can determine for more than finitely many pairs

(x, T) whether $C(x) \geq \phi(T)$. This shows that in general, although the undecidable statements based on Kolmogorov complexity are plentiful for each theory, we cannot explicitly construct them. Thus, the new approach entails loss of constructivity.

2.7.2 Barzdins's Lemma

Using Kolmogorov complexity one can quantify the distinction between recursively enumerable sets and recursive sets. Let A be a set of natural numbers.

Definition 2.7.1 The *characteristic sequence* of $A \subseteq \mathcal{N}$ is an infinite binary sequence $\chi = \chi_1 \chi_2 \ldots$ defined by

$$\chi_i = \begin{cases} 1 & \text{if } i \in A, \\ 0 & \text{otherwise.} \end{cases}$$

If A is recursively enumerable, and also its complement consisting of the i's such that $\chi_i = 0$ is recursively enumerable, then $f(i) = \chi_i$ is recursive, and the conditional complexity $C(\chi_{1:n}|n)$ is bounded by a fixed constant for all n. (The converse also holds, Exercise 2.3.4 on page 131.) But in the general case of recursively enumerable sets A, the complexity $C(\chi_{1:n}|n)$ can grow unboundedly with n. However, this growth is at best logarithmically slow, which shows that such characteristic sequences are very nonrandom. For instance, they are not random in the sense of Martin-Löf according to Theorem 2.5.4. The result below is known as *Barzdins's lemma.* (Actually, J.M. Barzdins proved the sharper version of Exercise 2.7.2 on page 180.)

Theorem 2.7.2 (i) *Any characteristic sequence χ of a recursively enumerable set A satisfies $C(\chi_{1:n}|n) \leq \log n + c$ for all n, where c is a constant dependent on A (but not on n).*

(ii) *Moreover, there is a recursively enumerable set such that its characteristic sequence χ satisfies $C(\chi_{1:n}) \geq \log n$ for all n.*

Proof. (i) Since A is recursively enumerable, there is a partial recursive function ϕ such that $A = \{x : \phi(x) < \infty\}$. Dovetail the computations of $\phi(1), \phi(2), \ldots$. In this way, we enumerate A without repetitions in the order in which the computations of the $\phi(i)$'s terminate. The prefix $\chi_{1:n}$ can be reconstructed from the number m of 1's it contains. For if we know m, then it suffices to use ϕ to enumerate the elements of A until we have found m distinct such elements less than or equal to n. If the set of these elements is $B = \{a_1, a_2, \ldots, a_m\}$, then by assumption these are *all* elements in A that do not exceed n. Hence, from B we can reconstruct all 1's in $\chi_{1:n}$, and the remaining positions must be the 0's. In this way, we can reconstruct $\chi_{1:n}$, given n, from a description of ϕ and m. Since $m \leq n$ and $C(\phi) < \infty$, we have proved (i).

(ii) Let ϕ_0 be the additively optimal function ϕ_0 of Theorem 2.1.1. Define $\chi = \chi_1\chi_2\ldots$ by

$$\chi_i = \begin{cases} 1 & \text{if } \phi_0(i,i) = 0, \\ 0 & \text{if } \phi_0(i,i) \neq 0 \text{ or } \phi_0(i,i) = \infty. \end{cases}$$

Obviously, χ is the characteristic sequence of a recursively enumerable subset of the natural numbers. We prove that χ satisfies the property stated in the theorem. For suppose to the contrary that $C(\chi_{1:n}) < \log n$ for some n. This means that there is a short program p, of length less than $\log n$, that computes $\chi_{1:n}$. Since $p < n$ this implies that $\phi_0(p,p) = \chi_p$, which contradicts the definition of χ_p. □

The converse of Theorem 2.7.2, Item (i), does not hold in general. This follows by the construction of a meager set that is not recursively enumerable. For instance, let χ be the characteristic sequence of a set A and $C(\chi_{1:n}|n) \geq n - c$ for infinitely many n and a fixed constant c. By Theorem 2.5.5 such strings are abundant, and by Theorem 2.7.2, Item (i), we find that A is not recursively enumerable. Construct a sequence ζ by $\zeta = \chi_1\alpha_1\chi_2\alpha_2\ldots$ with $\alpha_i = 0^{f(i)}$, where $f(i)$ is some fast-growing total recursive function with an inverse. Obviously, if ζ is the characteristic sequence of set B, then B is not recursively enumerable. But also there is now another constant c such that $C(\zeta_{1:n}|n) \leq f^{-1}(n) + c$ for all n. Choosing f such that $\log\log f(n) = n$ gives

$$C(\zeta_{1:n}|n) \leq \log\log n + O(1).$$

Theorem 2.7.2, Item (i), cannot be improved to the unconditional "$C(\chi_{1:n}) \leq \log n + c$ for all n and some c," since all χ's satisfying this are recursive (and hence the corresponding sets A are recursive) by Exercise 2.3.4 on page 131. Theorem 2.7.2, Item (ii), cannot be improved to the conditional "$C(\chi_{1:n}|n) \geq \log n$ for all n" by Exercise 2.7.3 on page 181.

Example 2.7.2 Diophantine equations are algebraic equations of the form $X = 0$, where X is built up from nonnegative integer variables and nonnegative integer constants by a finite number of additions $(A + B)$ and multiplications $(A \times B)$. The best-known examples are $x^n + y^n = z^n$, where $n = 1, 2, \ldots$.

Pierre de Fermat (1601–1665) stated that this equation has no solution in positive integers x, y, and z for n an integer greater than 2. (For $n = 2$ there exist solutions, for instance $3^2 + 4^2 = 5^2$.) However, he did not supply a proof of this assertion, often called *Fermat's last theorem*. After 350 years of withstanding concerted attempts to come up with a proof or disproof, the problem had become a celebrity among unsolved mathematical problems. However, A. Wiles [*Ann. of Math.*, 141:3(1995), 443–551] has finaly settled the problem by proving Fermat's last theorem. Let us for the moment disregard Wiles's proof and reason naively. Suppose we substitute all possible values for x, y, z with $x + y + z \leq n$, for $n = 3, 4, \ldots$. In this way, we recursively enumerate all solutions of Fermat's equation. Hence, such a process will eventually give a counterexample to Fermat's conjecture *if one exists*, but the process will never yield conclusive evidence if the conjecture happens to be true.

In his famous address to the International Congress of Mathematicians in 1900, D. Hilbert proposed twenty-three mathematical problems as a program to direct the mathematical efforts in the twentieth century. The tenth problem asks for an algorithm that given an arbitrary Diophantine equation, produces either an integer solution for this equation or indicates that no such solution exists. After a great deal of preliminary work by other mathematicians, the Russian mathematician Yu.V. Matijasevich finally showed that no such algorithm exists. Suppose we weaken the problem as follows. First, effectively enumerate all Diophantine equations, and consider the characteristic sequence $\Delta = \Delta_1 \Delta_2 \ldots$, defined by $\Delta_i = 1$ if the ith Diophantine equation is solvable, and 0 otherwise. Then $C(\Delta_{1:n}) \leq n + O(1)$. But the theorem above shows that $C(\Delta_{1:n}|n) \leq \log n + c$, for some fixed constant c. The nonrandomness of the characteristic sequence means that the solvability of Diophantine equations is highly interdependent—it is impossible for a random sequence of them to be solvable and the remainder unsolvable. ◇

Example 2.7.3 In the proof of Theorem 2.7.2, Item (ii), we used a set recursively isomorphic to the halting set $K_0 = \{\langle x, y\rangle : \phi_x(y) < \infty\}$. In fact, almost every recursively enumerable set that is complete under the usual reducibilities would have done. We can use this to obtain natural examples for incompressible finite strings. Let d be the number of elements in K_0 that are less than 2^n. Then by running all the computations of all $\phi_x(y)$ with $z < 2^n, z = \langle x, y\rangle$, in parallel until d of them have halted, we effectively find all computations among them that halt. That is, if χ is the characteristic sequence of K_0, then we can effectively compute $\chi_{1:m}$ (where $m = 2^n$) from d. The program p for this computation has length $l(p) \leq l(d) + c \leq n + c$, for some fixed constant c independent of n. By Theorem 2.7.2, Item (ii), the shortest program from which we can compute $\chi_{1:m}$ has length at least n. Hence, there is another constant c such that p is c-incompressible. ◇

Exercises

2.7.1. [10] Show that there exists a constant c such that $C(0^n|n) \leq c$ for all n, and $C(0^n) \geq \log n - c$ for infinitely many n.

2.7.2. • [26] Let $A \subseteq \mathcal{N}$ be a recursively enumerable set, and let $\chi = \chi_1\chi_2\ldots$ be its characteristic sequence. We use the uniform complexity $C(\chi_{1:n}; n)$ of Exercise 2.3.2 on page 130.

(a) Show that $C(\chi_{1:n}; n) \leq \log n + O(1)$ for all A and n.

(b) Show that there exists an A such that $C(\chi_{1:n}; n) \geq \log n$ for all n.

Comments. This implies Theorem 2.7.2. It is the original Barzdins's lemma. Source: J.M. Barzdins, *Soviet Math. Dokl.*, 9(1968), 1251–1254.

2.7.3. [27] Is there a symmetric form of Theorem 2.7.2 (Barzdins's lemma) using only conditional complexities? The answer is negative. Show that there is no recursively enumerable set such that its characteristic sequence χ satisfies $C(\chi_{1:n}|n) \geq \log n + O(1)$ for all n.

Comments. Hint: let χ be the characteristic sequence of a recursively enumerable set A. Consider $C(\chi_{1:f(n)}|f(n))$, with $\chi_{1:f(n)}$ containing exactly 2^{2^n} ones. Then, $\log f(n) \geq 2^n$. But using the partial recursive function enumerating A, we can compute $\chi_{1:f(n)}$, given $f(n)$, from just the value of n. Hence, we have a program of $\log n + O(1)$ bits for $\chi_{1:f(n)}$. Compare this to Barzdins's lemma (Theorem 2.7.2) and Exercise 2.7.2. Source: R.M. Solovay, sci.logic electronic newsgroup, 24 November 1989.

2.7.4. [34] Is there a symmetric form of Theorem 2.7.2 (Barzdins's lemma) using only unconditional complexities? The answer is negative. Show that there is a recursively enumerable set $A \subseteq \mathcal{N}$ and a constant c such that its characteristic sequence χ satisfies $C(\chi_{1:n}) \geq 2 \log n - c$ for infinitely many n.

Comments. First note the easy fact that the Kolmogorov complexity of $\chi_{1:n}$ is at most $2 \log n + O(1)$ for all n ($\leq \log n$ bits to specify n and $\leq \log n$ bits to specify $k = \sum_{i=1}^{n} \chi_i$). Hint: partition $\mathcal{N}$ into exponentially increasing half-open intervals $I_k = (t_k, 2^{t_k}]$ with $t_0 = 0$. Note that $\log(2^{t_k} - t_k) = 2 \log t_{k+1} - 2 - o(1)$ for $k \to \infty$. Use increasingly precise approximations of $C(\chi_{1:n})$ for $n \in I_k$ for increasing k to enumerate A. Source: R.M. Solovay, sci.logic electronic newsgroup, 24 November 1989; posed as open problem [O39] in the first printing of this book; solved by M. Kummer [*SIAM J. Comput.*, 25:6(1996), 1123–1143].

2.7.5. [25] Prove the following strange fact (Kamae's theorem). For every natural number m there is a string x such that for all but finitely many strings y, $C(x) - C(x|y) > m$.

Comments. There exist strings x such that almost all strings y contain a large amount of algorithmic information about x. Hint: x must be such that almost all large numbers contain much information about x. Let c be a large enough fixed constant. Let A be a recursively enumerable set of integers, and let $\alpha_1 \alpha_2 \ldots$ be the characteristic sequence of A. Set $x = x(k) = \alpha_1 \alpha_2 \ldots \alpha_h$, where $h = 2^k$. By Barzdins's lemma, Theorem 2.7.2, we can assume $C(x(k)) \geq k$. Enumerate A without repetition as $b_1, b_2, \ldots$. Let $m(k) = \max\{i : b_i \leq 2^k\}$. Then for any integer $y \geq m(k)$ we have $C(x(k)|y) \leq \log k + c$. Namely, using y we can enumerate $b_1, b_2, \ldots, b_t$, and with $\log k$ extra information describing k we can find $x(k)$. Therefore, $C(x) - C(x|y) \geq k - \log k - c$. Source: T. Kamae, *Osaka J. Math.*, 10(1973), 305–307. See also Exercise 2.2.13.

2.7.6. [25] Consider an enumeration $W_1, W_2, \ldots$ of all recursively enumerable sets. A simple set A is *effectively simple* if there is a recursive function f such that $W_i \subseteq \bar{A}$ implies that $d(W_i) \leq f(i)$ (where $\bar{A}$ is the complement of A). The set $\bar{A}$ is called *effectively immune.*

(a) Show that the set B defined in Theorem 2.7.1 is effectively simple.

(b) Show that the set RAND defined by $\{x : C(x) \geq l(x)\}$ is effectively immune.

(c) Show that Item (a) implies that B is Turing complete for the recursively enumerable sets.

Comments. Hint: the proof of Theorem 2.7.1, Item (iii), showing that B is simple actually shows that B is also effectively simple, which demonstrates Item (a). For Item (c), see for example P. Odifreddi, *Classical Recursion Theory*, North-Holland, 1989.

2.7.7. [32] Let $\phi_1, \phi_2, \ldots$ be the standard enumeration of partial recursive functions. The *diagonal halting set* is $\{x : \phi_x(x) < \infty\}$ (also denoted by K). The *Kolmogorov set* is $\{(x, y) : C(x) \leq y\}$. We assume familiarity with notions in Exercise 1.7.16. To say that a set A is *recursive in* a set B is the same as saying that A is Turing reducible to B.

(a) Show that the diagonal halting set is recursive in the Kolmogorov set.

(b) Show that the Kolmogorov set is recursive in the diagonal halting set.

(c) Show that the Kolmogorov set is Turing-complete for the recursively enumerable sets.

Comments. This means that if we can solve the halting problem, then we can compute C, and conversely. Hint for Item (a): given x we want to know whether $x \in K$, that is, whether $T_x(x)$ halts. Let $l(\langle x, T_x \rangle) = n$. Now use the Kolmogorov set to recursively find the least number t such that for all y with $l(y) = 2n$ and $C(y) < 2n$ the reference universal machine U computes y from some program of length less than $2n$ in at most t steps. Note that t is found with some organized dovetailing. Claim: $T_x(x)$ halts iff $T_x(x)$ halts within t steps (hence we can see whether $T_x(x)$ halts). If not, then we can use $T_x(x)$ as a clock and run the same dovetailing process as above, but now we produce a string of complexity $2n$ via a description of length n. Source: Attributed to P. Gács by W. Gasarch, personal communication February 13, 1992.

2.7.8. [42] We can express the nonrecursivity of $C(x)$ in terms of $C(C(x)|x)$, which measures what we may call the *complexity of the complexity function.* Denote $l(x)$ by n.

(a) Prove the *upper bound* $C(C(x)|x)) \leq \log n + O(1)$.

(b) Prove the following *lower bound*: For each length n there are strings x such that

$$C(C(x)|x) \geq \log n - \log\log n - O(1).$$

Comments. This means that x only marginally helps to compute $C(x)$; most information in $C(x)$ is extra information related to the halting problem. Hint for Item (b): same proof as in Section 3.8. Source: P. Gács, *Soviet Math. Dokl.*, 15(1974), 1477–1480.

2.7.9. [44] Show that every infinite sequence is Turing-reducible (Exercise 1.7.16, page 43, with sets replaced by characteristic sequences of sets) to an infinite sequence that is random with respect to the uniform measure.

Comments. C.H. Bennett raised the question whether every infinite binary sequence can be obtained from an incompressible one by a Turing machine. He proved this for a special case. Philosophically, the result implied in the exercise allows us to view even very pathological sequences as the result of two relatively well understood notions, to wit, the completely chaotic outcome of coin-tossing and a Turing machine transducer algorithm. Source: P. Gács, *Inform. Contr.*, 70(1986), 186–192. See also [W. Merkle, N. Mihailovic, *J. Symb. Logic*, 69(2004), 862–878].

2.7.10. [30] This exercise assumes knowledge of the notion of *Turing degree*, Exercise 1.7.16. Every Turing degree contains a set A such that if χ is the characteristic sequence of A, then $C(\chi_{1:n}|n) \leq \log n$ for all n.

Comments. Hence, a high degree of unsolvability of a set does not imply a high Kolmogorov complexity of the associated characteristic sequence. Hint: call a set B *semirecursive* if there exists a recursive linear ordering $<_B$ of $\mathcal{N}$ such that there exists a lower cut element y such that $B = \{x : x \leq_B y\}$. For any set A there is a semirecursive set B such that $B \equiv_T A$ [C.G. Jockusch, *Trans. AMS* 131(1968), 420–436]. Every semirecursive set B has a characteristic sequence χ of $(\mathcal{N}, <_B)$ such that $C(\chi_{1:n}|n) \leq \log n + c$, by the same proof as Theorem 2.7.2, Item (i). Since $<_B$ is recursive, the same property holds for the usual characteristic sequence of B. Source: W. Gasarch, Letter, August 1988. See also R.P. Daley, *J. Comput. System Sci.*, 9(1974), 151–163; *Math. Systems Theory*, 9(1975), 83–94; *Inform. Contr.*, 44(1980), 236–244.

2.7.11. [42] Use Kolmogorov complexity to prove the existence of Turing degrees of unsolvability (Exercise 1.7.16) between the recursive sets and Turing-complete sets (such as K_0).

Comments. Source: R.P. Daley, *J. Symb. Logic*, 46(1981), 460–474; *Inform. Contr.*, 52(1982), 52–67.

2.7.12. [39] We assume familiarity with the notion of truth-table reducibility. Let χ be the characteristic sequence of a recursively enumerable set A. Here $C(\chi_{1:n}; n)$ is the uniform complexity of Exercise 2.3.2.

(a) Show that A is complete under weak truth-table reducibility iff for some unbounded total recursive function $f(n)$, we have $C(\chi_{1:n}; n) \geq f(n)$.

(b) Show that A is complete under Turing reducibility iff $C(\chi_{1:n}; n) \geq f(n)$ for some unbounded total function f recursive in A.

Comments. For resource-bounded versions of Kolmogorov complexity the situation is quite different. Source: M.I. Kanovich, *Soviet Math. Dokl.*, 10(1969), 700–701; 11(1970), 1224–1228.

2.7.13. [20] Define the *state complexity* $S(x)$ of a finite binary string x as the least n such that there is a Turing machine with n states that started in the standard initial conditions of empty tape and distinguished start state will eventually halt with x on its output tape. All machines considered are of the original model as in Section 1.7. Define $B = \{\langle x, y\rangle : S(x) \leq y\}$.

(a) Prove that B is recursively enumerable but not recursive.

(b) Prove that B is Turing complete (in the sense of Exercise 1.7.16).

Comments. Suppose our Turing machines use an m-letter alphabet. Let $T_m(x)$ denote the complexity of x in terms of the minimal number of internal states of a Turing machine. Then

$$T_m(x) \sim C(x)/\,(m-1)\log C(x).$$

Source: problem by J. Andrews, electronic news, June 24, 1988; solutions by V.R. Pratt, R.M. Solovay, electronic news, June 1988.

2.7.14. [22] Show that the set K_0 used in Lemma 1.7.6 on page 35 is not many-to-one reducible to the set B featured in Corollary 2.7.2 on page 176, while B is many-to-one reducible to K_0.

Comments. Hint: Use Exercise 1.7.16. K_0 is m-complete, while B is simple and hence not m-complete. The set K_0 is of a *higher degree of unsolvability with respect to many-to-one reducibility* than B.

2.7.15. [32] Show that there exists an immune set I (a set without an infinite recursively enumerable subset, for instance the complement of a set B as in Theorem 2.7.1, Item (iii)), such that there is a probabilistic machine that computes the characteristic function of some infinite subset of I.

Comments. Hint: Use Theorems 2.5.4, 2.7.1, and the following framework. A *probabilistic* machine is just like a deterministic machine except that at some steps there are several actions (instead of a single action) that the machine can perform with given probabilities. For simplicity assume that there are exactly two possible actions, each with probability $\frac{1}{2}$. (That is, at each such choice the machine flips a coin.) That a probabilistic machine computes a function ϕ with probability p means that the machine with input x halts with output $\phi(x)$ with probability p. We usually assume $p > \frac{1}{2}$. It can be shown that a machine with any value p between zero and one can be simulated by a machine with value p close to one. It turns out that ϕ is computable by a probabilistic machine iff ϕ is partial recursive [K. de Leeuw, et al., pp. 183–212 in: *Automata Studies*, C.E. Shannon and J. McCarthy, eds., Princeton Univ. Press, 1956]. This result is often interpreted as showing that probabilistic machines cannot perform tasks that are impossible for deterministic machines. But a task may not consist only in finding an unambiguous value, but may consist in finding some value out of a set of possible values. In this form there are obviously tasks that deterministic machines cannot do that probabilistic machines can do, such as the construction of a nonrecursive sequence or to output the characteristic function of some infinite subset of a fixed immune set. The probabilistic machine computes such a characteristic sequence or set if it outputs the sequence or set with positive probability. Source: A.K. Zvonkin and L.A. Levin, *Russ. Math. Surv.*, 25:6(1970), 83–124, attributed to J.M. Barzdins.

2.7.16. [37] A set H of natural numbers is called *hyperimmune* if there is no total recursive function f such that $f(i) > h_i$ for all i, where h_i is the ith element of H in increasing order. That is, H is immune (Exercise 2.7.15, page 184) but the variety of immunity of H is due to the fact that the function that enumerates H's elements in increasing order of size grows faster than any recursive function. Prove the following:

(a) Every hyperimmune set H contains an infinite subset whose characteristic sequence is not computable by a probabilistic machine.

(b) However, there is a probabilistic machine that computes the characteristic sequence of some hyperimmune set.

Comments. Source: A.K. Zvonkin and L.A. Levin, *Russ. Math. Surv.*, 25:6(1970), 83–124, attribute Item (a) to V.N. Agafonov and L.A. Levin, and Item (b) to N.V. Petri. Hint: Item (a) follows from the fact that if a fixed set is computable by a probabilistic machine then it is recursively enumerable. Theorems 2.2 and 2.3 in P. Gács, [*Theoret. Comput. Sci.*, 22(1983), 71–93], cover related issues.

2.7.17. [19] We define a variant of the busy beaver function $BB(n)$ in Exercise 1.7.19 on page 45. Let $BC(n)$ be the largest natural number m such that $C(m) \leq n$. Let $\phi_1, \phi_2, \ldots$ be the standard effective enumeration of partial recursive functions.

(a) Show that $BC(n) > \phi(n)$, where $\phi = \phi_k$, for all $n \geq C(k) - 4 \log n$.

(b) Show that $BC(n)$ is not a recursive function.

(c) Show that the nonrecursiveness of $BC(n)$ can be used to prove the unsolvability of the halting problem (Lemma 1.7.5 on page 34) and vice versa.

(d) Let F be an axiomatizable sound formal theory that can be described completely (axioms, inference rules, ...) in m bits. Show that no provable true statement in F asserts "$BC(n) = x$" for $BC(n) = x$ with any $n > m + O(1)$.

Comments. Hint for Item (a): $C(\phi(n)) \leq C(k, n) + O(1)$. Then, $C(\phi(n)) \leq n - \log n$. Hence, $BC(n) > \phi(n)$. Hint for Item (b): It grows faster than any recursive function. Hint for Item (c): If the halting problem were solvable, we could compute $BC(n)$ from the outputs of all halting programs of length at most n. Conversely, every halting program p halts within $BC(n)$ steps, for $n \geq l(p) + O(1)$. So recursiveness of BC implies the solvability of the halting problem. This exercise is an application of Theorem 2.3.1, Item (iii). In fact, $BC(n)$ is some sort of inverse function of $m(n)$, the greatest monotonic increasing function bounding $C(n)$ from below. Source: G.J. Chaitin, pp. 108–111 in: *Open Problems in Communication and Computation*, T.M. Cover, B. Gopinath, eds., Springer-Verlag, 1988.

2.8 Algorithmic Information Theory

One interpretation of the complexity $C(x)$ is as the quantity of information needed for the recovery of an object x from scratch. Similarly, the conditional complexity $C(x|y)$ quantifies the information needed to recover x given only y. Hence the complexity is 'absolute information' in an object. Can we obtain similar laws for complexity-based 'absolute information theory' as we did for the probability-based information theory of Section 1.11?

If $C(x|y)$ is much less than $C(x)$, then we may interpret this as an indication that y contains a lot of information about x.

Definition 2.8.1 The *algorithmic information* about y contained in x is defined as

$$I_C(x : y) = C(y) - C(y|x).$$

Choosing reference function ϕ_0 in Theorem 2.1.1 with $\phi_0(x, \epsilon) = x$ yields

$$C(x|x) = 0 \text{ and } I_C(x : x) = C(x).$$

By the additive optimality of ϕ_0, these equations hold up to an additive constant independent of x, for any reference function ϕ_0. In this way we can view the complexity $C(x)$ as the algorithmic information about itself contained in an object. 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 Section 1.11.

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, Theorem 2.1.1. Intuitively, it is reasonable to require that

$$I_C(x : y) \geq 0,$$

up to an additive fixed constant independent of x and y. Formally, this follows easily from the definition of $I_C(x : y)$, by noting that $C(y) \geq C(y|x)$ up to an independent additive constant.

2.8.1 Entropy, Information, and Complexity

The major point we have to address is the relation between the Kolmogorov complexity and Shannon's entropy as defined in Section 1.11. Briefly, classic information theory says that a random variable X distributed according to $P(X = x)$ has entropy (complexity) $H(X) = \sum P(X = x) \log 1/P(X = x)$, where the interpretation is that $H(X)$ bits are on average sufficient to describe an outcome x. Algorithmic complexity says that an object x has complexity, or algorithmic information, $C(x)$ equal to the minimum length of a binary program for x. It is a beautiful fact that these two notions turn out to be much the same. The statement below may be called the theorem of equality between stochastic entropy and expected algorithmic complexity. (The theorem actually gives an inequality, but together with the simple argument in Example 2.8.1 on page 188 this turns into an asymptotic equality.)

Theorem 2.8.1 *Let $x = y_1 y_2 \ldots y_m$ be a finite binary string with $l(y_1) = \cdots = l(y_m) = n$. Let the frequency of occurrence of the binary representation of $k = 1, 2, 3, \ldots, 2^n$ as a y-block be denoted by $p_k = d(\{i : y_i = k\})/m$. Then up to an independent additive constant,*

$$C(x) \leq m(H + \epsilon(m)),$$

with $H = \sum p_k \log 1/p_k$, the sum taken for k from 1 to 2^n, and $\epsilon(m) = 2^{n+1} l(m)/m$. Note that $\epsilon(m) \to 0$ as $m \to \infty$ with n fixed.

Proof. Denote 2^n by N. To reconstruct x it suffices to know the number $s_k = p_k m$ of occurrences of k as a y_i in x, $k = 1, 2, \ldots, N$, together with x's serial number j in the ordered set of all strings satisfying these constraints. That is, we can recover x from $s_1, \ldots, s_N, j$. Therefore, up to an independent fixed constant, $C(x) \leq 2l(s_1) + \cdots + 2l(s_N) + l(j)$. By construction,

$$j \leq \binom{m}{s_1, \ldots, s_N},$$

a multinomial coefficient (Exercise 1.3.4 on page 10). Since also each $s_k \leq m$, we find that

$$C(x) \leq 2^{n+1} l(m) + l\binom{m}{s_1, \ldots, s_N}.$$

Writing the multinomial coefficient in factorials, and using Stirling's approximation, Exercise 1.5.4 on page 17, to approximate j, the theorem is proved. □

In Theorem 2.8.1 we have separated the frequency regularities from the remaining regularities. The entropy component mH measures the frequency regularities only, while the remaining component $m\epsilon(m)$ accounts for all remaining factors.

Example 2.8.1 For x representing the sequence of outcomes of independent trials, the inequality in Theorem 2.8.1 can be replaced by asymptotic equality with high probability.

We give a simple example to show the relation between the entropy H of a stochastic source X, emitting n-length binary words with probability $P(X = x)$ of outcome x, and the complexity C. Let $P(X = x) = 2^{-n}$ be the uniform probability distribution on the outcomes of length n. The entropy H in Theorem 2.8.1 is, according to Section 1.11, especially designed to measure frequency regularities. We show that it is asymptotically equal to the expected complexity of a string. By Theorem 2.2.1 almost all x are c-incompressible, that is, there are $2^n(1 - 2^{-c+1})$ many x's that have $C(x) \geq n - c$. A simple computation shows that the entropy $H(X) = \sum_{l(x)=n} P(X = x) \log 1/P(X = x)$ is asymptotically equal to the *expected complexity* $\mathbf{E} = \sum_{l(x)=n} P(X = x) C(x)$ of an n-length word. Namely, we obtain

$$\frac{n}{n + O(1)} \leq \frac{H(X)}{\mathbf{E}} < \frac{n}{(1 - 2^{-c+1})(n - c)}.$$

Substitute $c = \log n$ to obtain

$$\lim_{n \to \infty} \frac{H(X)}{\mathbf{E}} = 1.$$

◇

In Section 4.3.5 we prove the following generalization: Let p be a *recursive* probability distribution on $\mathcal{N}$, that is, there is a Turing machine computing the function p. Let, moreover, $K(x)$ be the prefix complexity as defined in Chapter 3. Then $\log 1/P(x)$ and $K(x)$ are close to each other with high probability. Since $|K(x) - C(x)| \leq 2 \log C(x)$ by Example 3.1.4 on page 203, also $\log 1/P(x)$ and $C(x)$ are close to each other with high probability.

In particular, the entropy $H = \sum_{l(x)=n} P(x) \log 1/P(x)$ of the distribution P is asymptotically equal to the expected complexity $\sum_{l(x)=n} P(x)K(x)$ of words of length n; see Section 8.1.1.

Because we saw a few lines above that $K(x)$ and $C(x)$ are equal up to a logarithmic additive term, the expected plain complexity $C(\cdot)$ is also asymptotically equal to the entropy,

$$\sum_x P(x)C(x) \sim \sum_x P(x) \log \frac{1}{P(x)}.$$

Thus, the intended interpretation of complexity $C(x)$ as a measure of the information content of an individual object x is supported by a tight quantitative relationship to Shannon's probabilistic notion.

2.8.2 Symmetry of Information

Is algorithmic information symmetric? In Section 1.11 we noted that in Shannon's information theory, the mutual information in one random variable about another one is symmetric. While the equation $I_C(x : y) = I_C(y : x)$ cannot be expected to hold exactly, a priori it can be expected to hold up to a constant related to the choice of reference function ϕ_0 in Theorem 2.1.1. However, with the current definitions, information turns out to be symmetric only up to a logarithmic additive term.

Example 2.8.2 By Theorem 2.2.1, there is a binary string x of each length n such that $C(x|n) \geq n$. Similarly, there are infinitely many n such that $C(n) \geq l(n)$. Choosing x such that its length n is random in this sense yields, up to independent constants,

$$\begin{aligned} I_C(x : n) &= C(n) - C(n|x) \geq l(n), \\ I_C(n : x) &= C(x) - C(x|n) \leq n - n = 0. \end{aligned}$$

◇

This example shows that the difference (the asymmetry of algorithmic information) $|I_C(x : y) - I_C(y : x)|$ can be of order the logarithm of the complexities of x and y. However, it cannot be greater, as we proceed to show now. This may be called the theorem of symmetry of algorithmic information for C-complexity. As usual, $C(x, y) = C(\langle x, y \rangle)$ is the length of the least program of U that prints out x and y and a way to tell them apart.

Theorem 2.8.2 *For all $x, y \in \mathcal{N}$, $C(x,y) = C(x) + C(y|x) + O(\log C(x,y))$.*

Since $C(x,y) = C(y,x)$ up to an additive constant term, the following symmetry of information property follows immediately.

Corollary 2.8.1 Up to an additive term $O(\log C(x,y))$,

$$C(x) - C(x|y) = C(y) - C(y|x).$$

Therefore,

$$|I_C(x:y) - I_C(y:x)| = O(\log C(x,y)).$$

Theorem 2.8.2 cannot be improved in general, since in Example 2.8.2 we have seen that the difference $|I_C(x:y) - I_C(y:x)|$ is at least $\log C(x)$ for some nontrivial x and y. The proof of Theorem 2.8.2 follows.

Proof. ($\leq$) We can describe $\langle x, y\rangle$ by giving a description of x, a description of y given x, and an indication of where to separate the two descriptions. If p is a shortest program for x and q is a shortest program for y, with $l(p) \leq l(q)$, then there is a Turing machine for which $\overline{l(p)}pq$ is a program to compute $\langle x, y\rangle$. Invoking the invariance theorem, Theorem 2.1.1, we obtain $C(x,y) \leq C(x) + C(y|x) + 2l(C(x)) + O(1)$.

($\geq$) Recall that the implied constant in the $O(\log C(x,y))$-notation can be both positive and negative. Thus, we need to prove that there is a constant $c \geq 0$ such that $C(x,y) \geq C(x) + C(y|x) - c \log C(x,y)$. Assume to the contrary that for every constant $c \geq 0$, there are x and y such that

$$C(y|x) > C(x,y) - C(x) + cl(C(x,y)). \tag{2.8}$$

Let $A = \{\langle u, z\rangle : C(u,z) \leq C(x,y)\}$. Given $C(x,y)$, the set A can be recursively enumerated. Let $A_x = \{z : C(x,z) \leq C(x,y)\}$. Given $C(x,y)$ and x, we have a simple algorithm to recursively enumerate the set A_x. One can describe y, given x, using its serial number in enumeration order of A_x and $C(x,y)$. Therefore,

$$C(y|x) \leq l(d(A_x)) + 2l(C(x,y)) + O(1). \tag{2.9}$$

By Equations 2.8, 2.9,

$$d(A_x) > 2^e, \quad e = C(x,y) - C(x) + (c-2)l(C(x,y)) - O(1). \tag{2.10}$$

But now we can obtain a too short description for x as follows. Given $C(x,y)$ and e, we can recursively enumerate the strings u that are candidates for x by satisfying

$$A_u = \{z : C(u,z) \leq C(x,y)\}, \tag{2.11}$$
$$2^e < d(A_u).$$

Denote the set of such u by U. Clearly, $x \in U$. Also,

$$\{\langle u, z\rangle : u \in U, z \in A_u\} \subseteq A. \tag{2.12}$$

The number of elements in A cannot exceed the available number of programs that are short enough to satisfy its definition:

$$d(A) \leq 2^{C(x,y)+O(1)}. \tag{2.13}$$

Combining Equations 2.11, 2.12, and 2.13, we obtain

$$d(U) \leq \frac{d(A)}{\min\{d(A_u) : u \in U\}} < \frac{d(A)}{2^e} \leq \frac{2^{C(x,y)+O(1)}}{2^e}.$$

Hence, we can reconstruct x from $C(x, y)$, e, and the serial number of x in enumeration order of U. Therefore,

$$C(x) < 2l(C(x, y)) + 2l(e) + C(x, y) - e + O(1).$$

Substituting e as given in Equation 2.10, this yields a contradiction, $C(x) < C(x)$, for large enough c. □

Exercises

2.8.1. [17] The following equality and inequality seem to suggest that the shortest descriptions of x contain some extra information besides the description of x.

(a) Show that $C(x, C(x)) = C(x) + O(1)$.

(b) Show that $C(x|y, i - C(x|y, i)) \leq C(x|y, i) + O(1)$.

Comments. These (in)equalities are in some sense pathological and may not hold for all reasonable descriptional complexities. However, these phenomena also hold for the prefix complexity K introduced in Chapter 3. Source: P. Gács, *Lecture Notes on Descriptional Complexity and Randomness*, Manuscript, Boston University, 1987.

2.8.2. [27] Let x be a string of length n.

(a) Show that the equality $C(x, C(x)) = C(C(x)|x) + C(x) + O(1)$ can be satisfied only to within an additive term of about $\log n$.

(b) Prove that $C(x, y) = C(x|y) + C(y)$ can hold only to within an additive logarithmic term without using Exercise 2.8.1, Item (a), and Exercise 2.7.8.

Comments. Hint for Item (a): use Exercise 2.8.1, Item (a), and Exercise 2.7.8. Hint for Item (b): additivity is already violated on random strings of random length. Source: P. Gács, *Ibid.*; A.K. Zvonkin and L.A. Levin, *Russ. Math. Surv.*, 25:6(1970), 83–124.

2.8.3. [12] Show that given x, y, and $C(x,y)$, one can compute $C(x)$ and $C(y)$ up to an additive logarithmic term $O(\log C(x,y))$.

Comments. Hint: use symmetry of information and upper semicomputability. Suggested by L. Fortnow.

2.8.4. [28] Let $\omega = \omega_1\omega_2\ldots$ be an infinite binary sequence. The entropy function $H(p)$ is defined by $H(p) = p\log 1/p + (1-p)\log 1/(1-p)$. Let $\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} \frac{1}{n}\omega_i = p$.

(a) Show that

$$C(\omega_{1:n}|n) \leq nH\left(\frac{1}{n}\sum_{i=1}^{n}\omega_i\right) + \log n + c.$$

(b) Prove the following: If the ω_i's are generated by coin flips with probability p for outcome 1 (a Bernoulli process with probability p), then for all $\epsilon > 0$,

$$\Pr\left\{\omega : \left|\frac{C(\omega_{1:n}|n)}{n} - H(p)\right| > \epsilon\right\} \to 0,$$

as n goes to infinity.

2.8.5. [26] Show that $2C(a,b,c) \leq C(a,b)+C(b,c)+C(c,a)+O(\log n)$.

Comments. For an application relating the 3-dimensional volume of a geometric object in Euclidean space to the 2-dimensional volumes of its projections, see the discussion in Section 6.13 on page 530. Hint: use the symmetry of information, Theorem 2.8.2. Source: D. Hammer and A.K. Shen, *Theor. Comput. Syst.*, 31:1(1998), 1–4.

2.9 History and References

The confluence of ideas leading to Kolmogorov complexity is analyzed in Section 1.8 through Section 1.12. Who did what, where, and when, is exhaustively discussed in Section 1.13. The relevant documents are dated R.J. Solomonoff, 1960/1964, A.N. Kolmogorov, 1965, and G.J. Chaitin, 1969. According to L.A. Levin, Kolmogorov in his talks used to give credit also to A.M. Turing (for the universal Turing machine). The notion of nonoptimal complexity (as a complexity based on shortest descriptions but lacking the invariance theorem) can be attributed, in part, also to A.A. Markov [*Soviet Math. Dokl.*, 5(1964), 922–924] and G.J. Chaitin [*J. ACM*, 13(1966), 547–569], but that is not a very crucial step from Shannon's coding concepts.

The connection between incompressibility and randomness was made explicit by Kolmogorov and later by Chaitin. Theorem 2.2.1 is due to Kolmogorov. The idea to develop an algorithmic theory of information is due to Kolmogorov, as is the notion of deficiency of randomness. Universal a priori probability (also based on the invariance theorem) is due to Solomonoff. This is treated in more detail in Chapter 4. (Solomonoff did not consider descriptional complexity itself in detail.)

In his 1965 paper, Kolmogorov mentioned the incomputability of $C(x)$ in a somewhat vague form: "[...] the function $C_\phi(x|y)$ cannot be effectively calculated (generally recursive) even if it is known to be finite for all x and y." Also Solomonoff suggests this in his 1964 paper: "it is clear that many of the individual terms of Eq. (1) are not 'effectively computable' in the sense of Turing [... but can be used] as the heuristic basis of various approximations." Related questions were considered by L. Löfgren [*Automata Theory*, E. Caianiello, ed., Academic Press, 1966, 251–268; *Computer and Information Sciences II*, J. Tou, ed., Academic Press, 1967, 165–175]. Theorem 1 in the latter reference demonstrates in general that for *every* universal function ϕ_0, $C_{\phi_0}(x)$ is not recursive in x. (In the invariance theorem we considered only universal functions using a special type of coding.)

Despite the depth of the main idea of Kolmogorov complexity, the technical expression of the basic quantities turned out to be inaccurate in the sense that many important relationships hold only to within an error term such as the logarithm of complexity. For instance, D.W. Loveland introduced n-strings in [*Inform. Contr.*, 15(1969), 510–526; *Proc. ACM 1st Symp. Theory Comput.*, 1969, 61–65] and proved that the length-conditional $C(x_{1:n}|n)$ measure is not monotonic in n, Example 2.2.5, page 119. He proposed a uniform complexity to solve this problem, and relationships between these complexities are the subject of Exercises 2.3.2, 2.3.4, 2.5.10, 2.5.11, 2.5.12, and 2.5.13.

In the subsequent development of this chapter we have used time and again the excellent 1970 survey by L.A. Levin and A.K. Zvonkin [*Russ. Math. Surv.*, 25:6(1970), 83–124], which describes mainly the research performed in the former USSR. We have drawn considerably on and profited from the point of view expressed in P. Gács's [*Komplexität und Zufälligkeit*, Ph.D. thesis, J.W. Goethe Univ., Frankfurt am Main, 1978, unpublished; *Lecture Notes on Descriptional Complexity and Randomness*, Manuscript, Boston University, 1987]. Another source for the Russian school is the survey by V.V. Vyugin, *Selecta Mathematica,* formerly *Sovietica*, 13:4(1994), 357–389 (translated from the Russian *Semiotika and Informatika*, 16(1981), 14–43).

In [A.K. Zvonkin and L.A. Levin, *Russ. Math. Surv.*, 25:6(1970), 83–124], Theorems 2.2.1 through 2.3.2 are attributed to Kolmogorov. The result on meager sets in Section 2.2 is from [M. Sipser, *Lecture Notes on Complexity Theory*, MIT Lab Computer Science, 1985, unpublished]. We avoided calling such sets 'sparse' sets because we need to reserve the term for sets that contain a polynomial number of elements for each length. The approximation theorem, Theorem 2.3.3, is stated in some form in [R.J. Solomonoff, *Inform. Contr.*, 7(1964), 1–22, 224–254], and is attributed also to Kolmogorov by Levin and Zvonkin. Some other properties of the integer function C we mentioned were observed by H.P. Katseff and M. Sipser [*Theoret. Comput. Sci.*, 15(1981), 291–309].

The material on random strings (sequences) in Sections 2.4 and 2.5 is primarily due to P. Martin-Löf [*Inform. Contr.*, 9(1966), 602–619; *Z. Wahrsch. Verw. Geb.*, 19(1971), 225–230]. The notions of random finite strings and random infinite sequences, complexity oscillations, lower semicomputable (sequential) Martin-Löf tests and the existence of universal (sequential) tests, the use of constructive measure theory, Theorems 2.4.2, and 2.5.1 through 2.5.5, are taken from P. Martin-Löf's papers. Weaker oscillations are mentioned by G.J. Chaitin [*J. ACM*, 16(1969), 145–159]. We also used [A.K. Zvonkin and L.A. Levin, *Russ. Math. Surv.*, 25:6(1970), 83–124; P. Gács, *Lecture Notes on Descriptional Complexity and Randomness*, Manuscript, Boston University, 1987].

As noted in the main text, the complexity oscillations of infinite sequences prevent a clear expression of randomness in terms of complexity. This problem was investigated by L.A. Levin in [A.K. Zvonkin and L.A. Levin, *Russ. Math. Surv.*, 25:6(1970), 83–124] and independently by C.P. Schnorr [*Lect. Notes Math.*, Vol. 218, Springer-Verlag, 1971]. As a part of the wider issue of (pseudo) random number generators and (pseudo) randomness tests, the entire issue of randomness of individual finite and infinite sequences is thoroughly reviewed by D.E. Knuth, *Seminumerical Algorithms*, Addison-Wesley, 1981, pp. 142–169; summary, history, and references: pp. 164–166. The whole matter of randomness of individual finite and infinite sequences of zeros and ones is placed in a wider context of probability theory and stochastics, and is analyzed in [A.N. Kolmogorov and V.A. Uspensky, *Theory Probab. Appl.*, 32(1987), 389–412; V.A. Uspensky, A.L. Semenov and A.K. Shen, *Russ. Math. Surv.*, 45:1(1990), 121–189; V.A. Uspensky, A.L. Semenov, An.A. Muchnik, A.L. Semenov, V.A. Uspensky, *Theoret. Comput. Sci.*, 2:207(1998), 1362–1376]. Developments in the theory, at the crossroads of notions of individual randomness, Kolmogorov complexity, and recursion theory have blossomed in the last decades. Such work has been partially incorporated in the main text, and in the exercises, of Chapters 2 through 4. Detailed treatment is beyond the scope and physical size of this book, and is the subject of more specialized treatment, as in R.G. Downey, D.R. Hirschfeldt, *Algorithmic Randomness and Complexity*, Springer-Verlag, New York, to appear; A.K. Shen, V.A. Uspensky, N.K. Vereshchagin, *Kolmogorov Complexity and Randomness*, Elsevier, Amsterdam, to appear; A. Nies, *Computability and Randomness*, Oxford Univ. Press, to appear.

Section 2.6, which analyzes precisely the relative frequencies of 0's and 1's and k-length blocks in individual infinite and finite sequences in terms of their Kolmogorov complexity, is from [M. Li and P.M.B. Vitányi, *Math. Systems Theory*, 27(1994), 365–376].

The recursion-theoretic properties we treat in Section 2.7 are related to Gödel's famous incompleteness theorem. Theorem 2.7.1 is attributed to J.M. Barzdins in [A.K. Zvonkin and L.A. Levin, *Russ. Math. Surv.*, 25:6(1970), 83–124]. The proof of Corollary 2.7.2 was given by G.J. Chaitin [*J. ACM*, 21(1974), 403–423; *Scientific American*, 232:5(1975), 47–52]. This application and some philosophical consequences have been advocated with considerable eloquence by G.J. Chaitin and C.H. Bennett [C.H. Bennett and M. Gardner, *Scientific American*, 241:5(1979), 20–34].

We also used the insightful discussion in [P. Gács, *Lecture Notes on Descriptional Complexity and Randomness*, Manuscript, Boston University, 1987]. These results are analyzed and critically discussed from a mathematical logic point of view by M. van Lambalgen [*J. Symb. Logic*, 54(1989), 1389–1400]. Theorem 2.7.2, Barzdins's lemma, occurs both in [J.M. Barzdins, *Soviet Math. Dokl.*, 9(1968), 1251–1254] and [D.W. Loveland, *Proc. ACM 1st Symp. Theory Comput.*, 1969, 61–65]. Examples in Section 2.7 are due to Kolmogorov, 1970, published much later as [*Russ. Math. Surv.*, 38:4(1983), 27–36] and a footnote in [L.A. Levin, *Problems Inform. Transmission*, 10:3(1974), 206–210].

The treatment of the relation between plain Kolmogorov complexity and Shannon's entropy in Section 2.8 is based on the work of A.N. Kolmogorov [*Problems Inform. Transmission*, 1:1(1965), 1–7; *IEEE Trans. Inform. Theory*, IT-14(1968), 662–665; *Russ. Math. Surv.*, 38:4(1983), 27–36; *Lect. Notes Math.*, Vol. 1021, Springer-Verlag, 1983, 1–5] and on [A.K. Zvonkin and L.A. Levin, *Russ. Math. Surv.*, 25:6(1970), 83–124]. The latter reference attributes Theorem 2.8.1 to Kolmogorov. Theorem 2.8.2 and its Corollary 2.8.1, establishing the precise error term in the additivity of complexity and symmetry of information as logarithmic in the complexity, are due to Levin and Kolmogorov [A.K. Zvonkin and L.A. Levin, *Russ. Math. Surv.*, 25:6(1970), 83–124; A.N. Kolmogorov, *Russ. Math. Surv.*, 38:4(1983), 27–36].