Theoretical Computer Science 501 (2013) 93–100

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Conditional Kolmogorov complexity and universal probability

Paul M.B. Vitányi

CWI, Science Park 123, 1098 XG Amsterdam, The Netherlands

ARTICLE INFO

Article history:
Received 23 January 2013
Received in revised form 16 May 2013
Accepted 12 July 2013
Communicated by O. Watanabe

Keywords:
Conditional Kolmogorov complexity
Conditional universal probability
Coding Theorem for conditional versions
Definitions of conditional probability

ABSTRACT

The Coding Theorem of L.A. Levin connects unconditional prefix Kolmogorov complexity with the universal semiprobability mass function. There are conditional versions referred to in several publications but as yet there exist no written proofs. Under the classic definition of conditional probability, there is no conditional version of the Coding Theorem. We give the appropriate definitions and provide complete proofs of the conditional version of the Coding Theorem.

1. Introduction

From the beginning of information theory we know that every probability mass function corresponds to a binary code. The code length equals roughly the logarithm of the inverse probability [11].[1] The binary code is a prefix code (no code element is a proper prefix of any other code element) and therefore uniquely decodable in one pass over the encoded message.

A *semiprobability* mass function is a function $p:\{0,1\}^* \to \mathcal{R}^+$ (the nonnegative real numbers) such that $\sum_x p(x) \leqslant 1$. (Note that $\{0,1\}^*$ is identified with the natural numbers according to the correspondence in Section 2.) L.A. Levin [7] proved that there is a greatest (up to a constant multiplicative factor) lower semicomputable semiprobability mass function. Denote such a function by $\mathbf{m}$. He furthermore proved that $\log 1/\mathbf{m}(x) = K(x) + O(1)$ where $K(x)$ is the prefix Kolmogorov complexity (the least upper semicomputable prefix code length). This is called the Coding Theorem (2). It is a central result in Kolmogorov complexity theory (a.k.a. algorithmic information theory). A conditional version of the Coding Theorem as referred to in [2,8,9,3,10] requires a function as follows:

Definition 1. Let x, y be finite binary strings. A function $\mathbf{m}(x|y)$ is a *universal conditional lower semicomputable semiprobability mass function* if it is (i) lower semicomputable; (ii) satisfies $\sum_x \mathbf{m}(x|y) \leqslant 1$ for every y; (iii) if $p(x|y)$ is a function satisfying (i) and (ii) then there is a constant c such that $c\mathbf{m}(x|y) \geqslant p(x|y)$ for all x and y.

There is no written complete proof of the conditional version of the Coding Theorem. Our aim is to provide such a proof and write it out in detail rather than rely on "clearly" or "obviously." One wants to be certain that applications of the conditional version of the Coding Theorem are well founded.

The necessary notions and concepts are given in appendices: Appendix A introduces prefix codes, Appendix B introduces Kolmogorov complexity, Appendix C introduces complexity notions, and Appendix D tells about the use of $O(1)$.

E-mail address: Paul.Vitanyi@cwi.nl.

[1] All logarithms are binary unless otherwise noted.

1.1. Related work

We can enumerate all lower semicomputable semiprobability mass functions with one argument. The arguments are elements of $\{0,1\}^*$ or natural numbers, Section 2. The enumeration list is denoted

$$\mathcal{P} = P_1, P_2, \ldots.$$

There is another interpretation possible. Let prefix Turing machine T_i be the ith element in the standard enumeration of prefix Turing machines $T_1, T_2, \ldots$. Then $Q_i(x) = \sum 2^{-|p|}$ where p is a program for T_i such that $T_i(p) = x$. This $Q_i(x)$ is the probability that prefix Turing machine T_i outputs x when the program on its input tape is supplied by flips of a fair coin. We can thus form the list

$$\mathcal{Q} = Q_1, Q_2, \ldots.$$

Both lists $\mathcal{P}$ and $\mathcal{Q}$ enumerate the same functions and there are computable isomorphisms between the two [9, Lemma 4.3.4].

Definition 2. If U is the reference universal prefix Turing machine, then the corresponding distribution in the Q-list is Q_U.

L.A. Levin [7] proved that

$$\mathbf{m}(x) = \sum_j \alpha_j P_j(x), \tag{1}$$

with $\sum_j \alpha_j \leqslant 1$, $\alpha_j > 0$, and α_j lower semicomputable, is a universal lower semicomputable semiprobability mass function. That is, obviously $\mathbf{m}$ is lower semicomputable and $\sum_x \mathbf{m}(x) \leqslant 1$. It is called a *universal* lower semicomputable semiprobability mass function since (i) it is itself a lower semicomputable semiprobability mass function and (ii) it multiplicatively (with factor α_j) dominates every lower semicomputable semiprobability mass function P_j.

Moreover, in [7] the Coding Theorem was proven

$$-\log \mathbf{m}(x) = -\log Q_U(x) = K(x), \tag{2}$$

where equality holds up to a constant additive term.

1.2. Results

We give a review of the classical definition of conditional probability versus the one used in the case of semicomputable probability. In Section 3 we show that the conditional version of (2) does not hold for the classic definition of conditional probability (Theorem 2). In Section 4 we consider Definition 1 of the conditional version of joint semicomputable semiprobability mass functions as used in [2,8,9,3,10]. For this definition the conditional version of (2) holds. We write all proofs out in complete detail.

2. Preliminaries

Let $x, y, z \in \mathcal{N}$, where $\mathcal{N}$ denotes the natural numbers and we identify $\mathcal{N}$ and $\{0,1\}^*$ according to the correspondence

$$(0, \epsilon), (1, 0), (2, 1), (3, 00), (4, 01), \ldots.$$

Here ϵ denotes the *empty word*. A *string* x is an element of $\{0,1\}^*$. The *length* $|x|$ of x is the number of bits in x, not to be confused with the absolute value of a number. Thus, $|010| = 3$ and $|\epsilon| = 0$, while $|-3| = |3| = 3$.

The emphasis is on binary sequences only for convenience; observations in any alphabet can be so encoded in a way that is 'theory neutral.' We use the natural numbers and the finite binary strings interchangeably.

3. Conditional probability

Let P be a probability mass function on sample space $\mathcal{N}$, that is, $\sum P(x) = 1$ where the summation is over $\mathcal{N}$. Suppose we consider $x \in \mathcal{N}$ and event $B \subseteq \mathcal{N}$ has occurred with $P(B) > 0$. According to Kolmogorov in [4] a new probability $P(x|B)$ has arisen satisfying:

1) $x \notin B$: $P(x|B) = 0$;
2) $x \in B$: $P(x|B) = P(x)/P(B)$;
3) $\sum_{x \in B} P(x|B) = 1$.

Let $\mathbf{m}$ be as defined in (1) with the sample space $\mathcal{N}$. Then $\sum \mathbf{m}(x) \leqslant 1$ and $\mathbf{m}$ is a semiprobability.

Definition 3. Define $\mathbf{m}'(x|B) = \mathbf{m}(x)/\mathbf{m}(B)$ and $\mathbf{m}(B) = \sum_{x \in B} \mathbf{m}(x)$.

Then, for $\mathbf{m}'(x|B)$ items 1) through 3) above hold.

Theorem 1. *Let $B \subseteq \mathcal{N}$. There exist x and B such that $-\log \mathbf{m}'(x|B) \neq K(x|B) + O(1)$.*

Proof. Since we use a special case of the event B in Theorem 2, the proof follows from that theorem. □

3.1. Lower semicomputable joint semiprobability mass functions

We show there is also no equivalent of the Coding Theorem for the conditional version of $\mathbf{m}'$ based on lower semicomputable joint semiprobability mass functions. The sample space is $\mathcal{N} \times \mathcal{N}$. The conditional event B above can be instantiated by $B = \{(z, y): z \in \mathcal{N}\} \subseteq \mathcal{N} \times \mathcal{N}$. We use a standard pairing function $\langle\cdot,\cdot\rangle$ to map $\mathcal{N} \times \mathcal{N}$ onto $\mathcal{N}$. For example, $\langle i, j\rangle = \frac{1}{2}(i + j)(i + j + 1) + j$.

Definition 4. Let $x, y \in \mathcal{N}$ and $f(\langle x, y\rangle)$ be a lower semicomputable function on a single argument such that we have $\sum_{\langle x,y\rangle} f(\langle x, y\rangle) \leqslant 1$. We use these functions f to define the *lower semicomputable joint semiprobability mass functions* $P(x, y) = f(\langle x, y\rangle)$.

By the proof of the later Theorem 3, Stage 1, one can effectively enumerate the family of lower semicomputable joint semiprobability mass functions by

$$\mathcal{P} = P_1, P_2, \ldots.$$

(Not to be confused with the P's and $\mathcal{P}$-list of Section 1.1.) We can now define the *lower semicomputable joint universal probability* by

$$\mathbf{m}(x, y) = \sum_j \alpha_j P_j(x, y), \tag{3}$$

with $\sum_j \alpha_j \leqslant 1$, $\alpha_j > 0$ and lower semicomputable. Classically, for a joint probability mass function $P(x, y)$ with $x, y \in \mathcal{N}$ one defines the conditional version [1] by $P(x|y) = P(x, y)/\sum_z P(z, y)$. The equivalent form for $\mathbf{m}$ is

Definition 5.

$$\mathbf{m}'(x|y) = \frac{\mathbf{m}(x, y)}{\sum_z \mathbf{m}(z, y)}.$$

Here $\mathbf{m}'(x|y)$ is a special case of $\mathbf{m}'(x|B)$ of Definition 3 if we set $B = \{(z, y): z \in \mathcal{N}\}$. Further,

$$\begin{aligned}\mathbf{m}'(x|y) &= \frac{\mathbf{m}(x, y)}{\sum_z \mathbf{m}(z, y)} \\ &= \frac{\sum_j \alpha_j P_j(x, y)}{\sum_z \sum_j \alpha_j P_j(z, y)} \\ &= \frac{\sum_j \alpha_j P_j(x, y)}{\sum_j \alpha_j \sum_z P_j(z, y)}\end{aligned}$$

is the quotient of two lower semicomputable functions. It may not be semicomputable (not proved here). We show that there is no conditional Coding Theorem for $\mathbf{m}'(x|y)$.

Theorem 2. *Let $x, y \in \{0, 1\}^*$. For every positive integer n there is a y with $|y| = n$ such that $-\log \mathbf{m}'(x|y) \geqslant K(x|y) + \Omega(\log n)$.*

Proof. By (3) and the Coding Theorem we have $-\log \mathbf{m}(x, y) = K(\langle x, y\rangle) + O(1)$. Clearly, $K(\langle x, y\rangle) = K(x, y) + O(1)$. The marginal universal probability $\mathbf{m}_2(y)$ is given by $\mathbf{m}_2(y) = \sum_z \mathbf{m}(z, y) \geqslant \mathbf{m}(\epsilon, y)$. Thus, with the last equality due to the Coding Theorem: $-\log \mathbf{m}_2(y) \leqslant -\log \mathbf{m}(\epsilon, y) = K(\langle \epsilon, y\rangle) + O(1) = K(y) + O(1)$. Since $\mathbf{m}'(x|y) = \mathbf{m}(x, y)/\mathbf{m}_2(y)$ by Definition 5, we have $-\log \mathbf{m}'(x|y) = -\log \mathbf{m}(x, y) + \log \mathbf{m}_2(y) \geqslant -\log \mathbf{m}(x, y) + \log \mathbf{m}(\langle \epsilon, y\rangle) = K(x, y) - K(y) + O(1) = K(x|y, K(y)) + O(1)$. Here the first inequality follows from the relation between $\mathbf{m}_2(y)$ and $\mathbf{m}(\langle \epsilon, y\rangle)$, while the last equality follows from $K(x, y) = K(y) + K(x|y, K(y)) + O(1)$ by the symmetry of information (8). In [2] it is shown that for every x and n there is a y with $|y| = n$ such that $K(x|y, K(y)) = K(x|y) + \Omega(\log n)$. □

4. Lower semicomputable conditional semiprobability

We consider lower semicomputable conditional semiprobabilities directly in order to obtain a conditional semiprobability that satisfies Definition 1.

Theorem 3. *There is a universal conditional lower semicomputable semiprobability mass function. We denote the reference such function by* $\mathbf{m}(x|y)$.

Proof. We prove the theorem in two stages. In Stage 1 we show that the two-argument lower semicomputable functions which sum over the first argument to at most 1 can be effectively enumerated as

$$P_1, P_2, \ldots .$$

This enumeration contains all and only lower semicomputable conditional semiprobability mass functions. In Stage 2 we show that P_0 as defined below multiplicatively dominates all P_j:

$$P_0(x|y) = \sum_j \alpha_j P_j(x|y),$$

with the α_j's satisfying certain requirements. Stage 1 consists of two parts. In the first part, we start from a standard enumeration of partial recursive functions, and transform them to enumerate all lower semicomputable two-argument functions; and in the second part we effectively change the lower semicomputable two-argument functions to functions that sum to at most 1 over the first argument. Such a change leaves the functions that were already conditional lower semicomputable semiprobability mass functions unchanged.

Stage 1 Start with a standard enumeration of all and only partial recursive functions $f_1, f_2, \ldots$. The partial recursive functions have a single natural number as argument and a single natural number as value. Using a standard pairing function $\langle\cdot,\cdot\rangle$ on the natural numbers (for example as defined in Section 3) twice, we change the single argument into three natural numbers, and, using the same pairing function, change the value into $\langle p, q\rangle$ $(p, q \in \mathcal{N})$ which denotes the rational number p/q. The result is an effective enumeration $\phi'_1, \phi'_2, \ldots$ of three-argument rational-valued partial recursive functions. That is, for every index j we interpret $f_j(\langle\langle x, y\rangle, k\rangle) = \langle p, q\rangle$ as $\phi'_j(x, y, k) = p/q$. Without loss of generality, each such ϕ'_j is modified to a partial recursive function ϕ_j satisfying the following properties:

For all $x, y, k, j \in \mathcal{N}$, **if** $\phi_j(x, y, k) < \infty$, **then** $\phi_j(x, y, 1), \phi_j(x, y, 2), \ldots, \phi_j(x, y, k-1) < \infty$ **and** $\phi_j(x, y, h+1) \geqslant \phi_j(x, y, h)$ $(1 \leqslant h < k)$. This can be achieved by dovetailing the computation of $\phi'_j(x, y, 1), \phi'_j(x, y, 2), \ldots$ and assigning computed values in enumeration order of halting, while respecting $\phi_j(x, y, h+1) \geqslant \phi_j(x, y, h)$, to $\phi_j(x, y, 1), \phi_j(x, y, 2), \ldots$.

We define the ψ-functions by $\lim_{k\to\infty} \phi_j(x, y, k) = \psi_j(x, y)$. (If $\phi_j(x, y, h)$ is undefined for $h > k_0$ then $\lim_{k\to\infty} \phi_j(x, y, k) = \phi_j(x, y, k_0)$. Note that possibly $\psi_j(x, y)$ equals infinity.) The resulting ψ-list consists by construction of all and only lower semicomputable two-argument real-valued functions. The two arguments are natural numbers and the real value is the limit of a sequence of rational numbers.

Each function ϕ_j is used to construct a function P_j that sums to at most 1 over the first argument. In the algorithm below, the local variable array P_j contains the current approximation to the values of P_j at each stage of the computation. This is doable because the nonzero part of the approximation is always finite. The *dovetailing* in Step 0 means that at stage t we execute Step 1 through Step 3 with $j + k_j = t$ (for all $j > 0$ and $k_j \geqslant 0$).

```
Step 0: for $t := 1, 2, \ldots$ dovetail with $j + k_j := t$ for all $j > 0$ and $k_j \geqslant 0$.
Step 1: if $j = t$ $(k_j = 0)$ then $P_j(x|y) := 0$ for all $x, y \in \mathcal{N}$.
Step 2: if $0 < j < t$ $(k_j = t - j)$ then compute $\phi_j(1, 1, k_j), \ldots, \phi_j(k_j, k_j, k_j)$. {If any $\phi_j(x, y, k_j)$, $1 \leqslant x, y \leqslant k_j$, is undefined,
        then the existing values of $P_j$ do not change.}
Step 3: if for some $y$ $(1 \leqslant y \leqslant k_j)$ we have $\phi_j(1, y, k_j) + \cdots + \phi_j(k_j, y, k_j) > 1$ then the existing values of $P_j$ do not change
        else for $x, y := 1, \ldots, k_j$ set $P_j(x|y) := \phi_j(x, y, k_j)$ {Step 3 is a test of whether the new assignment of $P_j$-values
        satisfies the lower semicomputable conditional semiprobability mass function requirements}.
```

If for some $x, y, k \in \mathcal{N}$ the value $\phi_j(x, y, k-1) < \infty$ and $\phi_j(x, y, k) = \infty$ in Step 2, then the last assigned values of P_j do not change any more even though the computation goes on forever. Now P_j is computable. If ψ_j in the ψ-list already satisfied $\sum_x \psi_j(x, y) \leqslant 1$, then the test in Step 3 is always passed and $P_j(x|y) = \psi_j(x, y)$, for all $x, y, j \in \mathcal{N}$.

Executing this procedure on all functions in the list $\phi_1, \phi_2, \ldots$ yields an effective enumeration $P_1, P_2, \ldots$ of all and only lower semicomputable conditional semiprobability mass functions. "All" lower semicomputable conditional semiprobability mass functions occurred already in the ψ-list one or more times and the algorithm leaves them unchanged. The algorithm takes care that

$$\sum_x P_j(x|y) \leqslant 1$$

for all indexes j. Hence we have "only" lower semicomputable conditional semiprobability mass functions in the P-list.

STAGE 2 Define the function P_0 as

$$P_0(x|y) = \sum_{j \geqslant 1} \alpha_j P_j(x|y),$$

with α_j chosen such that $\sum_j \alpha_j \leqslant 1$, $\alpha_j > 0$ and lower semicomputable for all j. To fix thoughts we can set $\alpha_j = 2^{-j}$. Then P_0 is a conditional semiprobability mass function since

$$\sum_x P_0(x|y) = \sum_{j \geqslant 1} \alpha_j \sum_x P_j(x|y) \leqslant \sum_{j \geqslant 1} \alpha_j \leqslant 1.$$

The function $P_0(\cdot|\cdot)$ is also lower semicomputable, since $P_j(x|y)$ is lower semicomputable in x, y, j. The coefficients α_j are by definition lower semicomputable for all j. Finally, P_0 multiplicatively dominates each P_j since for all $x, y \in \mathcal{N}$ we have $P_0(x|y) \geqslant \alpha_j P_j(x|y)$ while $\alpha_j > 0$. Therefore, P_0 is a universal lower semicomputable conditional semiprobability mass function according to Definition 1.

We can choose the α_j's in the definition of P_0 in the proof above

$$\alpha_j = 2^{-K(j)}.$$

Then $\sum_j \alpha_j \leqslant 1$ by the ubiquitous Kraft inequality [6] (satisfied by the prefix complexity K), $\alpha_j > 0$ and it is lower semicomputable. We define $\mathbf{m}(x|y)$ by

$$\mathbf{m}(x|y) = \sum_{j \geqslant 1} 2^{-K(j)} P_j(x|y).$$

We call $\mathbf{m}(x|y)$ the *reference universal lower semicomputable conditional semiprobability mass function.* $\square$

Corollary 1. *If $P(x|y)$ is a lower semicomputable conditional semiprobability mass function, then $2^{K(P)}\mathbf{m}(x|y) \geqslant P(x|y)$, for all x, y. That is, $\mathbf{m}(x|y)$ multiplicatively dominates every lower semicomputable conditional semiprobability mass function $P(x|y)$.*

4.1. A priori probability

Let $P_1, P_2, \ldots$ be the effective enumeration of all lower semicomputable conditional semiprobability mass functions constructed in Theorem 3. There is another way to effectively enumerate all lower semicomputable conditional semiprobability mass functions similar to the unconditional $\mathcal{Q}$-list in Section 1.1. Let the input to a prefix machine T (with the string y on its auxiliary tape) be provided by an infinitely long sequence of fair coin flips. The probability of generating an initial input segment p is $2^{-|p|}$. If $T(p, y) < \infty$, that is, T's computation on p with y on its auxiliary tape terminates, then the machine T with y on its auxiliary tape, being a prefix machine, will read exactly p and no further.

Let $T_1, T_2, \ldots$ be the standard enumeration of prefix machines in [9]. For each prefix machine T, define

$$Q_T(x|y) = \sum_{T(p,y)=x} 2^{-|p|}. \tag{4}$$

In other words, $Q_T(x|y)$ is the probability that T with y on its auxiliary tape computes output x if its input is provided by fair coin flips. This means that for every string y we have that Q_T satisfies

$$\sum_{x \in \mathcal{N}} Q_T(x|y) \leqslant 1.$$

We can approximate $Q_T(\cdot|y)$ for every string y as follows. (The algorithm uses the local variable $Q(x)$ to store the current approximation to $Q_T(x|y)$.)

Step 1: Fix $y \in \{0,1\}^*$. Initialize $Q(x) := 0$ for all x.
Step 2: Dovetail the running of all programs on T with auxiliary y so that in stage k, step $k - j$ of program j is executed. Every time the computation of some program p halts with output x, increment $Q(x) := Q(x) + 2^{-|p|}$.

The algorithm approximates the displayed sum in (4) by the contents of $Q(x)$. Since $Q(x)$ is nondecreasing, this shows that Q_T is lower semicomputable. Starting from a standard enumeration of prefix machines $T_1, T_2, \ldots$, this construction gives for every $y \in \{0,1\}^*$ an enumeration of only lower semicomputable conditional probability mass functions

$$Q_1(\cdot|y), Q_2(\cdot|y), \ldots.$$

To merge the enumerations for different y we use dovetailing over the index j of $Q_j(\cdot|y)$ and y. The $\mathcal{P}$-list of Theorem 3 contains all elements enumerated by this Q-enumeration. In [9, Lemma 4.3.4] the reverse is shown.

Definition 6. The *universal conditional a priori semiprobability* on the positive integers (finite binary strings) is defined as

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

where U is the reference prefix machine.

Remark 1. To beat a dead horse again: the use of prefix Turing machines in the present discussion rather than plain Turing machines is necessary. By Kraft's inequality the series $\sum_p 2^{-|p|}$ converges (to $\leqslant 1$) if the summation is taken over all halting programs p of any fixed prefix machine with a fixed auxiliary input y. In contrast, if the summation is taken over all halting programs p of a plain Turing machine, then the series $\sum_p 2^{-|p|}$ might diverge.

4.2. The conditional Coding Theorem

Theorem 4. *There is a constant c such that for every x and y,*

$$\log \frac{1}{\mathbf{m}(x|y)} = \log \frac{1}{Q_U(x|y)} = K(x|y),$$

with equality up to an additive constant c.

Proof. Since $2^{-K(x|y)}$ represents the contribution to $Q_U(x|y)$ by a shortest program for x given the auxiliary y, we have $2^{-K(x|y)} \leqslant Q_U(x|y)$, for all x, y. Using the algorithm in Section 4.1 we know that $Q_U(x|y)$ is lower semicomputable. By the universality of $\mathbf{m}(x|y)$ in the class of lower semicomputable conditional semiprobability mass functions, $Q_U(x|y) = O(\mathbf{m}(x|y))$.

It remains to show that $\mathbf{m}(x|y) = O(2^{-K(x|y)})$. This is equivalent to proving that $K(x|y) \leqslant \log 1/\mathbf{m}(x|y) + O(1)$, as follows. Exhibit a prefix code E encoding each source word x, given y, as a code word $E(x|y)$ satisfying

$$\left|E(x|y)\right| \leqslant \log \frac{1}{\mathbf{m}(x|y)} + O(1),$$

together with a decoding prefix machine T such that $T(E(x|y), y) = x$. Then,

$$K_T(x|y) \leqslant \left|E(x|y)\right|,$$

and by the Invariance Theorem (6)

$$K(x|y) \leqslant K_T(x|y) + c_T,$$

with c_T a nonnegative constant that may depend on T but not on x, y. Note that T is fixed by the above construction. On the way to constructing E as required, we recall a construction related to that [11] for the Shannon–Fano code:

Lemma 1. *Let $x \in \{0,1\}^*$ and p be a function such that $\sum_x p(x) \leqslant 1$. There is a binary prefix code e such that the code words satisfy $|e(x)| \leqslant \log 1/p(x) + 2$.*

Proof. Let $[0,1)$ be the half-open real unit interval, corresponding to the sample space $S = \{0,1\}^\infty$. Each element ω of S corresponds to a real number $0.\omega$. Recall that x is a finite binary string. The half-open interval $[0.x, 0.x+2^{-|x|})$ corresponding to the cylinder (set) of reals $\Gamma_x = \{0.\omega : \omega = x\ldots \in S\}$ is called a *binary interval*. We cut off disjoint, consecutive, adjacent (not necessarily binary) intervals I_x of length $p(x)$ from the left end of $[0,1)$, $x = 1, 2, \ldots$. Let i_x be the length of the longest binary interval contained in I_x. Set $e(x)$ equal to the binary word corresponding to the leftmost such interval. Then $|e(x)| = \log 1/i_x$. It is easy to see that I_x is covered by at most four binary intervals of length at most i_x, from which the lemma follows. □

We use this construction to find a prefix machine T such that $K_T(x|y) \leqslant \log 1/\mathbf{m}(x|y) + c$. That $\mathbf{m}(x|y)$ is not computable but only lower semicomputable results in $c = 3$. Since $\mathbf{m}(x|y)$ is lower semicomputable and total, there is a recursive function $\phi(x, y, t)$ with $\phi(x, y, t) \leqslant \mathbf{m}(x|y)$ and $\phi(x, y, t+1) \geqslant \phi(x, y, t)$, for all t. Moreover, $\lim_{t\to\infty} \phi(x, y, t) = \mathbf{m}(x|y)$. Let $\psi(x, y, t)$ be the greatest recursive lower bound of the following special form on $\phi(x, y, t)$ defined by

$$\psi(x, y, t) := \left\{2^{-k}\colon\ 2^{-k} \leqslant \phi(x, y, t) < 2 \cdot 2^{-k} \text{ and } \phi(x, y, j) < 2^{-k} \text{ for all } j < t\right\},$$

and $\psi(x, y, t) := 0$ otherwise. Let ψ enumerate its range without repetition. Then, for every y we have

$$\sum_{x,t} \psi(x, y, t) = \sum_x \sum_t \psi(x, y, t) \leqslant \sum_x 2\mathbf{m}(x|y) \leqslant 2.$$

The series $\sum_t \psi(x,y,t)$ can converge to precisely $2\mathbf{m}(x|y)$ only in case there is a positive integer k such that $\mathbf{m}(x|y) = 2^{-k}$.

In a manner similar to the proof of Lemma 1 for every y we chop off consecutive, adjacent, disjoint half-open intervals $I_{x,y,t}$ of length $\psi(x,y,t)/2$, in enumeration order of a dovetailed computation of all $\psi(x,y,t)$, starting from the left-hand side of $[0,1)$. We have already shown that this is possible. It is easy to see that we can construct a prefix machine T as follows: Let $e \in \{0,1\}^*$ be such that Γ_e is the leftmost largest binary interval of $I_{x,y,t}$. Then $T(e,y) = x$. Otherwise, $T(e,y) = \infty$ (T does not halt).

By construction of ψ, for each pair x, y there is a t such that $\psi(x,y,t) > \mathbf{m}(x|y)/2$. Each interval $I_{x,y,t}$ has length $\psi(x,y,t)/2$. Each I-interval contains a binary interval Γ_e of length at least one-half of that of I (because the length of I is of the form 2^{-k}, it contains a binary interval of length 2^{-k-1}). Therefore, $2^{-|e|} \geqslant \mathbf{m}(x|y)/8$. We choose $E(x|y) = e$. This implies $K_T(x|y) \leqslant \log 1/\mathbf{m}(x|y) + 3$, which was what we had to prove. $\square$

Corollary 2. *The above result plus Corollary 1 give: Let P be a lower semicomputable conditional semiprobability mass function. There is a constant $c_P = K(P) + O(1)$ such that $K(x|y) \leqslant \log 1/P(x|y) + c_P$.*

5. Conclusion

The conditional version of the Coding Theorem of L.A. Levin, Theorem 4, requires a lower semicomputable conditional semiprobability that multiplicatively dominates all other lower semicomputable conditional semiprobabilities as in Theorem 3. The conventional form of the conditional (adapted to probabilities summing to at most 1) applied to the distribution (1) satisfying the original Coding Theorem (2) is false. This is shown by Theorems 1 and 2.

Acknowledgements

I thank Alexander Shen, Péter Gács, and the referee for commenting on the manuscript.

Appendix A. Self-delimiting code

A binary string y is a *proper prefix* of a binary string x if we can write $x = yz$ with $z \neq \epsilon$. A set $\{x, y, \ldots\} \subseteq \{0,1\}^*$ is *prefix-free* if for any pair of distinct elements in the set neither is a proper prefix of the other. A prefix-free set is also called a *prefix code* and its elements are called *code words*.

Appendix B. Kolmogorov complexity

For precise definitions, notation, and results see the text [9]. Kolmogorov complexity was introduced in [5] and the prefix Kolmogorov complexity we use in [7]. One realizes this complexity by considering a special type of Turing machine with a one-way input tape, one or more (a finite number) of two-way separate work tapes, and a one-way output tape. Such Turing machines are called *prefix* Turing machines. If a machine T halts with output x after having scanned all of p on the input tape, but not further, then $T(p) = x$ and we call p a *program* for T. It is easy to see that $\{p: T(p) = x, x \in \{0,1\}^*\}$ is a *prefix code*.

Let $T_1, T_2, \ldots$ be a standard enumeration of all prefix Turing machines with a binary input tape, for example the lexicographic length-increasing ordered prefix Turing machine descriptions [9]. Let $\phi_1, \phi_2, \ldots$ be the enumeration of corresponding prefix functions that are computed by the respective prefix Turing machines (T_i computes ϕ_i). These functions are the *partial recursive* functions or *computable* functions (of effectively prefix-free encoded arguments). We denote the function computed by a Turing machine T_i with p as input and y as conditional information by $\phi_i(p, y)$. One of the main achievements of the theory of computation is that the enumeration $T_1, T_2, \ldots$ contains a machine, say T_u, that is computationally universal and optimal in that it can simulate the computation of every machine in the enumeration when provided with its program and index in the sense that it computes a function ϕ_u such that $\phi_u(\langle i, p\rangle, y) = \phi_i(p, y)$ for all i, p, y. We fix one such machine, denote it by U and designate it as the *reference universal prefix Turing machine*.

Definition 7. The *conditional prefix Kolmogorov complexity* of x given y (as auxiliary information) *with respect to prefix Turing machine* T_i is

$$K_i(x|y) = \min\{|p|: \phi_i(p, y) = x\}. \tag{5}$$

The *conditional prefix Kolmogorov complexity* $K(x|y)$ is defined as the conditional Kolmogorov complexity $K_U(x|y)$ with respect to the reference prefix Turing machine U. The *unconditional* version is set to $K(x) = K(x|\epsilon)$.

By the definition we have that $\{K(x): x \in \mathcal{N}\}$ is a prefix-free set, and for every fixed $y \in \mathcal{N}$ the set $\{K(x|y): x \in \mathcal{N}\}$ is prefix-free. The prefix Kolmogorov complexity $K(x|y)$ satisfies the following so-called Invariance Theorem:

$$K(x|y) \leqslant K_i(x|y) + c_i \tag{6}$$

for all i, x, y, where c_i depends only on T_i (asymptotically, the reference machine is not worse than any other machine). Intuitively, $K(x|y)$ represents the minimal number of bits required to compute x from input y (provided the set of programs is prefix-free). As formulated in [3], "The functions $K(\cdot)$ and $K(\cdot|\cdot)$, though defined in terms of a particular machine model, are machine-independent up to an additive constant and acquire an asymptotically universal and absolute character through Church's thesis, and from the ability of universal machines to simulate one another and execute any effective process."

A prominent property of the prefix-freeness of $\{K(x)\colon\ x \in \mathcal{N}\}$ is that we can interpret $2^{-K(x)}$ as a semiprobability mass function. By the fundamental Kraft's inequality [6] (see for example [1,9]) we know that if $l_1, l_2, \ldots$ are the code-word lengths of a prefix code, then $\sum_x 2^{-l_x} \leqslant 1$. Hence,

$$\sum_x 2^{-K(x)} \leqslant 1. \tag{7}$$

This leads to the notion of universal semiprobability mass function $\mathbf{m}(x) = 2^{-K(x)}$ which we may view as a rigorous form of Occam's razor. Namely, the probability $\mathbf{m}(x)$ is great if x is simple ($K(x)$ is small such as $K(x) = O(\log|x|)$) and $\mathbf{m}(x)$ is small if x is complex ($K(x)$ is large such as $K(x) \geqslant |x|$).

The remarkable *symmetry of information* property was proven in [12] for the plain complexity version in less precise form, and more precisely in [2] for the prefix complexity version below. Denote $K(x,y) = K(\langle x,y\rangle)$. Then,

$$\begin{aligned} K(x,y) &= K(x) + K\big(y|x, K(x)\big) + O(1) \\ &= K(y) + K\big(x|y, K(y)\big) + O(1). \end{aligned} \tag{8}$$

By [2], for every $y, n \in \mathcal{N}$ there are $x \in \mathcal{N}$ with $|x| = n$ such that $K(y|x, K(x)) = K(y|x) + \Omega(\log n)$.

Appendix C. Computability notions

A pair of nonnegative integers, such as (p,q) can be interpreted as the rational p/q. We assume the notion of a computable function with rational arguments and values. A function $f(x)$ with x rational is *semicomputable from below* if it is defined by a rational-valued total computable function $\phi(x,k)$ with x a rational number and k a nonnegative integer such that $\phi(x,k+1) \geqslant \phi(x,k)$ for every k and $\lim_{k\to\infty}\phi(x,k) = f(x)$. This means that f (with possibly real values) can be computed in the limit from below (see [9, p. 35]). A function f is *semicomputable from above* if $-f$ is semicomputable from below. If a function is both semicomputable from below and semicomputable from above then it is *computable*.

We now consider a subclass of the lower semicomputable functions. A function $f: \mathcal{N} \to \mathcal{R}^+$ is a *semiprobability* mass function if $\sum_x f(x) \leqslant 1$ and it is a *probability* mass function if $\sum_x f(x) = 1$. Using a standard pairing function as in Section 3 we can define lower semicomputable functions over two arguments by setting $f(\langle x,y\rangle) = f(x,y)$. Such a function f is a *conditional semiprobability* mass function if $\sum_x f(x,y) \leqslant 1$.

Appendix D. Precision

It is customary in this area to use "additive constant c" or equivalently "additive $O(1)$ term" to mean a constant, accounting for the length of a fixed binary program, independent from every variable or parameter in the expression in which it occurs. In this paper we use the prefix complexity variant of Kolmogorov complexity. Prefix complexity of a string exceeds the plain complexity of that string by at most an additive term that is logarithmic in the length of that string.

References

[1] T.M. Cover, J.A. Thomas, Elements of Information Theory, Wiley & Sons, 1991.
[2] P. Gács, On the symmetry of algorithmic information, Soviet Math. Dokl. 15 (1974) 1477–1480; P. Gács, Soviet Math. Dokl. 15 (1974) 1480 (Corrigendum).
[3] P. Gács, Lecture notes on descriptional complexity and randomness, Manuscript, Computer Science, Boston University, Boston, MA, 2010, http://www.cs.bu.edu/faculty/gacs/papers/ait-notes.pdf.
[4] A.N. Kolmogorov, Grundbegriffe der Wahrscheinlichkeitsrechnung, Springer-Verlag, Berlin, 1933; English translation (by N. Morrison): Foundations of the Theory of Probability, Chelsea, 1956.
[5] A.N. Kolmogorov, Three approaches to the quantitative definition of information, Probl. Inf. Transm. 1 (1) (1965) 1–7.
[6] L.G. Kraft, A device for quantizing, grouping and coding amplitude modulated pulses, Master's thesis, Dept. of Electrical Engineering, MIT, Cambridge, MA, 1949.
[7] L.A. Levin, Laws of information conservation (non-growth) and aspects of the foundation of probability theory, Probl. Inf. Transm. 10 (1974) 206–210.
[8] L.A. Levin, On the principle of conservation of information in intuitionistic mathematics, Soviet Math. Dokl. (1976) 601–605.
[9] M. Li, P.M.B. Vitányi, An Introduction to Kolmogorov Complexity and Its Applications, 3rd edition, Springer-Verlag, 2008.
[10] A.K. Shen, V.A. Uspensky, N.K. Vereshchagin, Kolmogorov Complexity and Randomness, MUHMO, Moscow, 2013 (published in Russian by the Moscow Center for Continuous Mathematical Education).
[11] C.E. Shannon, The mathematical theory of communication, Bell Syst. Tech. J. 27 (1948) 379–423, 623–656.
[12] A.K. Zvonkin, L.A. Levin, The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms, Russian Math. Surveys 25 (6) (1970) 83–124.