Open access • Journal Article • DOI:10.1109/18.481775

Sequential codes, lossless compression of individual sequences, and Kolmogorov complexity — Source link

John C. Kieffer, En-hui Yang

Institutions: University of Minnesota, Nankai University

Published on: 01 Jan 1996 - IEEE Transactions on Information Theory (IEEE)

Topics: Kolmogorov structure function, Sequential decoding, Kolmogorov complexity, Lossless compression and Data compression

Sequential Codes, Lossless Compression of Individual Sequences, and Kolmogorov Complexity

John C. Kieffer, *Fellow, IEEE*, and En-hui Yang

***Abstract*—A general class of sequential codes for lossless compression of individual sequences on a finite alphabet is defined, including many types of codes that one would want to implement. The principal requirement for membership in the class is that the encoding and decoding operations be performable on a computer. The OPTA function for the class of codes is then considered, which is the function that assigns to each individual sequence the infimum of the rates at which the sequence can be compressed over this class of sequential codes. Two results about the OPTA function are obtained: 1) it is shown that any sequential code in the class compresses some individual sequence at a rate strictly greater than the rate for that sequence given by the OPTA function; and 2) it is shown that the OPTA function takes a value strictly greater than that of the Kolmogorov complexity rate function for some individual sequences.**

***Index Terms*—Lossless compression, individual sequences, sequential codes, Kolmogorov complexity, Lempel–Ziv algorithm.**

I. Introduction

THROUGHOUT the paper, we fix a finite set A containing at least two symbols; the set A shall serve as the alphabet for all information sources that are considered. We let α denote the number of symbols in A; we refer to the symbols in A as α-ary symbols. In this paper, we shall call an infinite sequence $x = (x_1, x_2, \cdots)$ of symbols from A an *individual sequence*, following the usage in [13] and subsequent papers. Suppose an information source generates some individual sequence. We shall be concerned with how well we can losslessly compress this individual sequence via sequential codes. We give here an informal description of the class of sequential codes that shall be allowed. We require that the first step in sequential encoding of an individual sequence be a parsing step in which the individual sequence is parsed into variable-length strings, each parsed string coming from a certain prefix set. Each variable-length string in the parsing is then replaced by a binary codeword from a prefix set—concatenation of these codewords yields the encoder output sequence in response to the individual sequence. Conceptually, decoding is done in much the same way as encoding; the encoded sequence is parsed, and each parsed string is replaced by the corresponding string of the individual sequence that was encoded. The encoder input and encoder output sequences of prefix sets that are employed evolve dynamically with time in a prescribed manner. Also, the encoding and decoding operations are required to be performable on a computer, that is, they are expressible as recursive functions. The resulting class of sequential codes shall be denoted Σ—this class of codes shall be formally laid out in Section II of the paper. The class Σ includes many types of codes of practical interest; for example, all finite-state codes [13] and the Lempel–Ziv code [7], [12] belong to this class.

When an individual sequence x is encoded by a sequential code $\sigma \in \Sigma$, the resulting asymptotic compression rate $\rho(x|\sigma)$ in code bits per α-ary symbol tells us how well the sequential code has performed. (Encode the first n α-ary symbols, count the number of code bits, divide by n, and take the limit superior as $n \to \infty$.) The optimum performance theoretically attainable in lossless compression of x via sequential codes is then the number $\rho(x)$, the infimum of $\rho(x|\sigma)$ over sequential codes $\sigma \in \Sigma$. Accordingly, we call the real-valued function $x \to \rho(x)$ *the OPTA function.*

The purpose of this paper is to resolve the following two questions concerning the OPTA function:

Question 1: Is there a universal sequential code? (A universal sequential code is a sequential code $\sigma \in \Sigma$ such that $\rho(x) = \rho(x|\sigma)$ for every individual sequence x.)

Question 2: Does the OPTA function coincide with the Kolmogorov complexity rate function?

Discussion of Question 1: Define a probability measure on the set of individual sequences to be *stationary* if it is preserved by the one-sided shift transformation $(x_1, x_2, \cdots) \to (x_2, x_3, \cdots)$. Then, the Lempel–Ziv code σ_{LZ} is "almost universal" with respect to the stationary probability measures, as the following fact (which is easily deduced from known results—see Appendix I) makes clear:

Fact 1.1: Let Ω_1 be the set of all individual sequences x such that $\rho(x|\sigma_{LZ}) = \rho(x)$. Then $\mu(\Omega_1) = 1$ for every stationary probability measure μ on the set of individual sequences.

However, the Lempel–Ziv code is not universal in the sense of this paper, i.e., providing the optimum compression performance for *every* individual sequence. To see this, suppose we take $A = \{0, 1\}$ for simplicity. Let x^* be the individual sequence obtained by concatenating together all binary strings of finite length, starting with the strings of length one, then those of length two, etc. Then $x^* = 0100011011000001 \cdots$.

Manuscript received February 11, 1994; revised March 7, 1995. This work was supported in part by the National Science Foundation under Grants NCR-9003106 and NCR-9304984. The material in this paper was presented in part at the IEEE International Symposium on Information Theory, Trondheim, Norway, 1994.

J. C. Kieffer is with the Department of Electrical Engineering, University of Minnesota, Minneapolis, MN 55455 USA.

E. Yang is with the Department of Mathematics, Nankai University, Tianjin 300071, P. R. China.

Publisher Item Identifier S 0018-9448(96)00564-0.

Those readers familiar with the Lempel–Ziv algorithm can easily see that x^* is not compressed by σ_{LZ}; that is, $\rho(x^*|\sigma_{LZ}) = 1$. Since there is an algorithm for generating x^*, one can construct a sequential code σ^* for which $\rho(x^*|\sigma^*) = 0$, and which compresses every other individual sequence at least as well as σ_{LZ} does; the existence of the code σ^* implies that σ_{LZ} is not universal. We shall later resolve Question 1 in the negative by duplicating the preceding argument for any sequential code in Σ. Any such code possesses a program which tells how it operates. This program proves to be the basis for the code's undoing in the sense that one is able to construct from its program another program which generates a sequence that is not compressed by the given code.

Discussion of Question 2: For each individual sequence $x = (x_1, x_2, \cdots)$, define $\overline{\boldsymbol{K}}(x)$ to be the limit superior as $n \to \infty$ of $\boldsymbol{K}(x_1, x_2, \cdots, x_n)/n$, where $\boldsymbol{K}(x_1, x_2, \cdots, x_n)$ is the Kolmogorov complexity of the string $(x_1, x_2, \cdots, x_n)$. The function $x \to \overline{\boldsymbol{K}}(x)$ is called the *Kolmogorov complexity rate function*. It is easy to show (and we prove this later; see Theorem 2, Section IV) that the Kolmogorov complexity rate function lower-bounds the OPTA function. It is natural then to investigate to what extent the Kolmogorov complexity rate function and the OPTA function coincide. The following fact (also easily deducible from known results—see Appendix I) gives us the manner in which these two functions "almost" coincide.

Fact 1.2: Let Ω_2 be the set of all individual sequences x such that $\rho(x) = \overline{\boldsymbol{K}}(x)$. Then $\mu(\Omega_2) = 1$ for every stationary probability measure μ on the set of individual sequences.

We describe another result of this type. Let the terminology A-string denote any finite sequence of symbols from A. Let N denote the set of natural numbers $\{1, 2, 3, \cdots\}$. We say that a probability measure μ on the set of individual sequences is *computable* if there is a Turing machine for which both of the following are true:

1) The machine accepts as input any pair (y, n) in which y is an A-string and $n \in N$.
2) For each input (y, n) the machine generates an output $(r, s) \in N \times N$ such that the μ-probability of the set of individual sequences that start with y lies strictly between $r/s - n^{-1}$ and $r/s + n^{-1}$.

Then, we have the following result, whose proof is sketched in Appendix I.

Fact 1.3: The set Ω_2 has measure one with respect to every computable probability measure on the set of individual sequences.

In view of Facts 1.2 and 1.3, Question 2 is natural. Theorem 3 of Section IV (the main result of this paper) provides an emphatic "No" to Question 2 in the sense that there exist individual sequences x for which $\overline{\boldsymbol{K}}(x) = 0$ while $\rho(x) = \log_2 \alpha$.

II. A Class of Sequential Codes

In this section, we make clear the notion of sequential code and, in particular, what it means for a sequential code to be a member of the class Σ.

A. Notation and Terminology

If S is a set and $n \in N$, then S^n denotes the set $\{(s_1, \cdots, s_n) : s_i \in S, 1 \le i \le n\}$ consisting of all strings of length n from S. The notation S^* shall denote the set of all strings of finite length that can be formed from the symbols in S. It is convenient to include in S^* the empty string (the string containing no symbols), which shall be denoted by λ_S. Thus $S^* = \{\lambda_S\} \cup S \cup S^2 \cup \cdots$. If $y \in S^*$, then $|y|$ denotes the length of y; note that $|\lambda_S| = 0$. We let S^∞ denote the set of all infinite sequences $(s_1, s_2, \cdots)$ whose entries are chosen from S.

If $u_1, u_2, \cdots, u_m$ is a finite sequence of strings in S^*, then $u_1 * u_2 * \cdots * u_m$ shall denote the string in S^* formed by writing down the symbols in u_1, followed by the symbols in u_2, etc.; the string $u_1 * u_2 * \cdots * u_m$ is called the *concatenation* of the strings $u_1, u_2, \cdots, u_m$.

If $s = (s_1, \cdots, s_m) \in S^m$ and $n \le m$, then s^n denotes the string $(s_1, \cdots, s_n)$. Similarly, if $s = (s_1, s_2, \cdots) \in S^\infty$, then s^n denotes the string $(s_1, \cdots, s_n)$.

Given two strings $y_1, y_2 \in S^*$, we say y_1 is a *prefix* of y_2 if there is a string $u \in S^*$ such that $y_2 = y_1 * u$, and we say that y_1 is a *proper prefix* of y_2 if y_1 is a prefix of y_2 and $y_1 \neq y_2$. Also, we say that y_1 is a *suffix* of y_2 if there is a string u such that $y_2 = u * y_1$. We say that $y \in S^*$ is a prefix of an infinite sequence s of symbols from S if $y = \lambda_S$ or if $y = s^n$ for some $n \in N$. If $u_1, u_2, \cdots$ is an infinite sequence of strings from S^*, then the notation $u = u_1 * u_2 * \cdots$ shall denote that $u \in S^\infty$ is the sequence for which $u_1 * u_2 * \cdots u_m$ is a prefix for every $m \in N$.

A subset P of S^* is said to be a *prefix set* if $y_1 = y_2$ whenever y_1, y_2 are members of P in which y_1 is a prefix of y_2, and is said to be *complete* if each sequence in S^∞ has a prefix in P. (For example, $\{0, 10, 11\}$ is a complete prefix subset of $\{0, 1\}^*$.)

The notation $f: S_1 \to S_2$ shall denote that f is a function which takes its values in the set S_2 and whose domain is a subset of S_1. This is a departure from the usual convention in which the notation $f: S_1 \to S_2$ implies that S_1 is the domain of f; we have departed from conventional usage in order for us to accomodate recursive functions (see below).

Let S_i be the set N, the set S^*, or a finite Cartesian product of sets of these two types, $i = 1, 2$. A function $f: S_1 \to S_2$ is said to be *recursive* if there exists a Turing machine [3, ch. 3] such that for every $y \in S_1$ in the domain of f, the machine halts and prints $f(y)$ in response to input y, and for every $y \in S_1$ not in the domain of f, the machine does not halt in response to input y. A recursive function $f: S_1 \to S_2$ is said to be *total recursive* if its domain is all of S_1. The class of recursive functions has a useful property which, for the purposes of this paper, shall be called the closure property. The closure property states that a function built up from performing finitely many operations on recursive functions is itself recursive, provided that each of the operations used is one of three basic operations called composition, primitive recursion, and minimization [3, ch. 14]. We shall sometimes use the closure property to conclude that a given function is recursive without having to show the

existence of a Turing machine that computes the values of the function.

All logarithms in this paper shall be to base two.

B. Concept of Parsing Rule

A *parsing rule* π is defined to be a sequence of pairs $\pi = \{(W_n, \Psi_n): n \in N\}$ such that the following two properties are valid:

3) Each W_n is a finite complete prefix subset of A^*, where A is the alphabet mentioned in Section I.
4) Each Ψ_n is a mapping from W_n into N.

Let $\pi = \{(W_n, \Psi_n): n \in N\}$ be a parsing rule and let x be any individual sequence. If one applies the parsing rule π to x, one obtains the sequence of A-strings $u_1, u_2, \cdots$ and the sequence of positive integers $i_1, i_2, \cdots$ such that

5) The string u_1 is the member of W_1 such that x starts with u_1; $i_1 = 1$.
6) For $t \geq 2$, $i_t = \Psi_{i_{t-1}}(u_{t-1})$ and u_t is the member of W_{i_t} such that $u_1 * u_2 * \cdots * u_t$ is a prefix of x.

The sequence of strings $\{u_1, u_2, \cdots\}$ shall be called the π-parsing or parsing of x. The individual strings u_i in the parsing shall be called phrases.

A parsing rule shall be employed as the basis for a sequential code, as follows. One first employs a device, called a parser, for accomplishing the task of parsing an individual sequence into the phrases according to a given parsing rule. Then, the phrases in the parsing are encoded into binary codewords one-by-one, and these codewords are then concatenated to form the encoded sequence which is the output of the sequential code in response to the given individual sequence. The precise manner in which codewords are assigned to the phrases in a parsing shall be described later in this section.

We now give some parsing rules commonly used as building blocks for sequential codes.

Example 1: The k-block parsing rule is the one in which each individual sequence is parsed into phrases all of which have length k. To illustrate, the 3-block parsing of $x = 010010111001\cdots$ starts with the phrases 010, 010, 111, 001.

Example 2: The Lempel–Ziv parsing of this same x starts with the phrases 0, 1, 00, 10, 11, 100. Each phrase in the parsing is a new phrase (a phrase that has not been previously listed).

It shall be convenient for us to introduce also the concept of the parsing of a string of finite length. Let $\pi = \{(W_n, \Psi_n): n \in N\}$ be a parsing rule. Let $y \in A^*$ and let $n \in N$. Informally, the (π, n)-parsing of y is the finite sequence of A-strings that results when y is fed into the parser with the parser in initial state n. Formally, this concept is defined as follows:

7) If y starts with no member of W_n, then the (π, n)-parsing of y is taken to be $\{\lambda_A\}$.
8) If y starts with a member of W_n, generate A-strings $\{u_1, u_2, \cdots, u_t\}$ (where $t \geq 1$) and integers $i_1, i_2, \cdots, i_{t+1}$ according to the rules
   i) $i_1 = n$; $i_k = \Psi_{i_{k-1}}(u_{k-1})$, $2 \leq k \leq t+1$.
   ii) $u_k \in W_{i_k}$ $(1 \leq k \leq t)$.
   iii) y starts with $u_1 * u_2 * \cdots * u_t$.
   iv) There is no $u \in W_{i_{t+1}}$ such that y starts with $u_1 * u_2 * \cdots * u_t * u$.

Then $\{u_1, u_2 \cdots, u_t\}$ is the (π, n)-parsing of y, and the final parsing state is defined to be i_{t+1}; the strings appearing in the (π, n)-parsing are termed the phrases of the parsing. We say that the (π, n)-parsing $\{u_1, \cdots, u_t\}$ of y is *complete* if $y = u_1 * u_2 * \cdots * u_t$. Finally, we take the π-parsing of y to be the (π, n)-parsing of y with $n = 1$.

C. Parsing Delay

Suppose the successive symbols in an individual sequence x are generated at times $t = 1, 2, \cdots$, respectively. A given parsing rule π is applied to this stream of symbols. Suppose the symbol of x generated at time $t = n$ occurs as the initial symbol in a phrase of the π-parsing of x, and suppose this phrase is of length k_n. Then, one has to wait until the symbols at times $t = n+1, n+2, \cdots, n+k_n-1$ are generated in order to determine the termination point of the phrase; if one encodes the individual sequence phrase by phrase, a delay of $k_n - 1$ time units has thus been introduced from time $t = n$ until encoded information about the $t = n$ symbol can be transmitted to the decoder. To control this parsing delay, a reasonable requirement to make is that k_n/n converge to zero as $n \to \infty$.

We now formalize the requirement on parsing delay alluded to at the end of the preceding paragraph. We do this using the concept of the delay modulus of a parsing rule. If π is a parsing rule, then its delay modulus is the function $\omega_\pi: N \to [0, \infty]$ defined for each $Q \in N$ by

$$\omega_\pi(Q) \triangleq \sup_{q \geq Q,\, x \in A^\infty} \frac{|u_{q+1}(x)|}{|u_1(x)| + |u_2(x)| + \cdots + |u_q(x)|}$$

where $\{u_i(x): i \in N\}$ denotes the π-parsing of the individual sequence x.

In defining sequential codes, we shall want to employ parsing rules π in which the parsing delay is controlled by requiring that $\omega_\pi(Q) \to 0$ as $Q \to \infty$.

Example 1: For the k-block parsing rule π, it is easy to see that

$$\omega_\pi(Q) = \frac{1}{Q}, \quad Q \in N$$

and hence, the requirement $\omega_\pi(Q) \to 0$ as $Q \to \infty$ is satisfied.

Example 2: For the Lempel–Ziv parsing rule π, one can check that

$$\omega_\pi(Q) \leq \frac{\beta}{\log(Q+1)}, \quad Q \in N$$

for some constant β, $0 < \beta < \infty$. The requirement $\omega_\pi(Q) \to 0$ as $Q \to \infty$ is satisfied.

D. Sequential Code Concept

We define a *sequential code* to be a sequence of triples $\sigma = \{(W_n, \Psi_n, \Phi_n): n \in N\}$ such that the following two properties are satisfied:

9) $\{(W_n, \Psi_n): n \in N\}$ is a parsing rule.

10) For each $n \in N$, Φ_n is a one to one mapping of W_n onto a prefix subset of $\{0, 1\}^*$.

The parsing rule given by Property 9) shall be called the parsing rule underlying σ; it shall be denoted by π_σ. We describe how a sequential code $\sigma = \{(W_n, \Psi_n, \Phi_n): n \in N\}$ is used to encode an individual sequence x into a sequence $y \in \{0, 1\}^\infty$. First, the π_σ-parsing $\{u_t: t \in N\}$ of x is generated. Let $\{i_t: t \in N\}$ be the resulting sequence of parser states defined by

$$i_1 = 1; \quad i_t = \Psi_{i_{t-1}}(u_{t-1}) \ (t \geq 2). \tag{2.1}$$

Then

$$y = \Phi_{i_1}(u_1) * \Phi_{i_2}(u_2) * \Phi_{i_3}(u_3) \cdots .$$

If σ is a sequential code and $y \in A^*$, then $L(\sigma, y)$ shall denote the total of the lengths of the binary codewords assigned by σ to the phrases in the parsing of y according to the parsing rule underlying σ. The rate at which the sequential code σ compresses the individual sequence x is the nonnegative extended real number $\rho(x|\sigma)$ defined by

$$\rho(x|\sigma) \triangleq \limsup_{n\to\infty} \frac{L(\sigma, x^n)}{n}.$$

We now define Σ to be the class of codes consisting of every sequential code $\sigma = \{(W_n, \Psi_n, \Phi_n): n \in N\}$ for which the following four properties are satisfied:

Property 2.1: The delay modulus $\omega_{\pi_\sigma}(Q) \to 0$ as $Q \to \infty$.

Property 2.2: There is a total recursive function $F: N \times A^* \to N$ such that

$$F(n, y) = 1, \quad y \in W_n$$
$$F(n, y) = 2, \quad \text{otherwise.}$$

Property 2.3: There is a recursive function $G: N \times A^* \to N$ such that

$$G(n, y) = \Psi_n(y), \quad \text{for every } n, y \text{ in which } y \in W_n.$$

Property 2.4 There is a recursive function $H: N \times A^* \to \{0, 1\}^*$ such that

$$H(n, y) = \Phi_n(y), \quad \text{for every } n, y \text{ in which } y \in W_n.$$

The OPTA function ρ alluded to in Section I can now be formally defined as the function $\rho: A^\infty \to [0, \infty]$ in which

$$\rho(x) = \inf\{\rho(x|\sigma): \sigma \in \Sigma\}, \quad x \in A^\infty.$$

The following are some examples of sequential codes in Σ.

Example 1: A sequential code $\sigma = \{(W_n, \Psi_n, \Phi_n): n \in N\}$ is called a *block code* if, for some k, $m \in N$, $W_n = A^k$, and $\Phi_n = \Phi_{k,m}$ for all n, where $\Phi_{k,m}$ is a mapping from A^k into $\{0, 1\}^m$. Any block code is a member of Σ. Furthermore, it is easy to construct a sequence of block codes $\{\sigma_n\}$ such that $\rho(x|\sigma_n) \to \log \alpha$ for any individual sequence x. Thus the OPTA function ρ takes its values in the interval $[0, \log \alpha]$.

Example 2: A sequential code $\sigma = \{(W_n, \Psi_n, \Phi_n): n \in N\}$ is said to be a *finite-state* sequential code if there is $m \in N$ such that, whenever $\{u_t: t \in N\}$ is the parsing of an individual sequence according to the parsing rule underlying σ, and $\{i_t: t \in N\}$ is the sequence generated according to (2.1), then $i_t \leq m$ for all t. Any finite-state sequential code is a member of Σ.

Example 3: The Lempel–Ziv code σ_{LZ} is a sequential code in Σ which bears a special relationship to the class of finite-state sequential codes. Ziv and Lempel [13] proved that the following statement holds for any individual sequence x:

$$\rho(x|\sigma_{LZ}) \leq \inf\{\rho(x|\sigma): \sigma \text{ a finite-state sequential code}\}.$$

The above examples indicate that the class of sequential codes Σ includes many types of sequential codes of practical significance. We therefore feel justified in concentrating on the class Σ in the sequel.

III. Nonexistance of Universal Sequential Codes

We define an individual sequence x to be *recursive* if there exists a total recursive function $f: N \to A^*$ such that $f(n) = x^n$, $n \in N$. Recursive sequences can be compressed well by sequential codes. In fact, it is easy to show that if an individual sequence x' is recursive, then there is a sequential code $\sigma' \in \Sigma$ such that $\rho(x'|\sigma') = 0$. Given any sequential code $\sigma \in \Sigma$, we shall show how to construct a recursive individual sequence x' which is not compressed by σ in the sense that $\rho(x'|\sigma) \geq \log \alpha$. Since, as discussed above, some other sequential code in Σ will compress x' well, this suggests the possibility that a sequential code in Σ can be constructed which outperforms the given code σ in the sense of compressing every individual sequence at least as well as does σ, while compressing some individual sequences (including x') better than does σ. We combine all of these thoughts in the following theorem, which shall be proved in the present section.

Theorem 1: Let $\sigma \in \Sigma$ be arbitrary. Then there exists a recursive individual sequence x' and $\sigma' \in \Sigma$ such that

i) $\rho(x'|\sigma) \geq \log \alpha$.
ii) $\rho(x'|\sigma') = 0$.
iii) $\rho(x|\sigma') \leq \rho(x|\sigma)$ for any individual sequence x.

The following result is an immediate corollary of Theorem 1.

Corollary 1: There exists no universal sequential code in the class Σ. That is, there exists no $\sigma \in \Sigma$ such that $\rho(x|\sigma) = \rho(x)$ for every individual sequence x.

Before we go into the details of the proof of Theorem 1, let us introduce a useful partial ordering of sequential codes.

A. A Partial Ordering of Sequential Codes

We define the concept (useful to us later on) of what it means for one sequential code to be subordinate to another sequential code. Let $\sigma = \{(W_n, \Psi_n, \Phi_n): n \in N\} \in \Sigma$. We say that $\sigma' = \{(W'_n, \Psi'_n, \Phi'_n): n \in N\} \in \Sigma$ is *subordinate to* σ if there is a total recursive function $g: N \to N$ with $g(1) = 1$ such that if $n \in N$ and $y \in W'_n$, the following are true:

1) The $(\pi_\sigma, g(n))$-parsing of y is complete with final state $g(\Psi'_n(y))$.

2) $\Phi'_n(y) = \Phi_{i_1}(u_1) * \Phi_{i_2}(u_2) * \cdots * \Phi_{i_t}(u_t)$, where $\{u_1, \cdots, u_t\}$ is the $(\pi_\sigma, g(n))$-parsing of y and $\{i_1, \cdots, i_t\}$ are the parser states

$$i_1 = g(n); \quad i_k = \Psi_{i_{k-1}}(u_{k-1}), \quad 2 \le k \le t.$$

As the preceding definition is somewhat abstract, it is helpful to have in mind an informal idea of what it means for σ' to be subordinate to σ. The phrases in the parsing of an individual sequence according to the parsing rule $\pi_{\sigma'}$ (call these σ'-phrases) are related to the phrases in the parsing of that same sequence via π_σ (call these σ-phrases) in the following way: The σ-phrases are partitioned into groups of phrases each consisting of finitely many phrases, and then each group of σ-phrases gives rise to a σ'-phrase via concatenation. Each σ'-phrase is then encoded into the binary codeword obtained by concatenating together the codewords for the σ-phrases in the group of σ-phrases that gave rise to that σ'-phrase.

The relation "is subordinate to" is a partial order on Σ. We shall occasionally make use of cofinal subsets with respect to this partial order. We say that a subset Σ' of Σ is *cofinal* if for any $\sigma \in \Sigma$, there exists $\sigma' \in \Sigma'$ such that σ' is subordinate to σ.

Three facts about the partial order "is subordinate to" shall be needed in the sequel. We state these facts here. The first of these facts follows easily from the definition given above:

Fact 3.1: If σ' is subordinate to σ, then

$$\rho(x|\sigma') \le \rho(x|\sigma), \quad x \in A^\infty.$$

For technical reasons, we occasionally need to make use of sequential codes in which the lengths of the phrases in the parsing of any individual sequence "blow up" sufficiently rapidly. In particular, we shall deal with a particular class of such sequential codes—we define Σ' to be the subset of Σ consisting of all $\sigma' \in \Sigma$ satisfying the following property:

Property 3.1: The rth phrase in the $\pi_{\sigma'}$-parsing of any individual sequence is of length at least $\sqrt{r}$, $r \in N$.

Our remaining two facts detail useful attributes of the class of sequential codes Σ'. Fact 3.2 is proved in Appendix II. Fact 3.3 is a simple consequence of Facts 3.1 and 3.2, which we have stated separately for emphasis.

Fact 3.2: The class of codes Σ' is a cofinal subset of Σ.

Fact 3.3: For any individual sequence x,

$$\rho(x) = \inf\{\rho(x|\sigma): \sigma \in \Sigma'\}.$$

The proof of Theorem 1 shall also hinge upon the following lemma.

Lemma 1: Let W be a complete prefix subset of A^*. Let Φ be a one-to-one map from W onto a prefix subset of $\{0, 1\}^*$. Then there exists $u \in W$ such that

$$|\Phi(u)| \ge |u| \log \alpha. \tag{3.1}$$

Proof of Lemma 1: Because W is a complete prefix set, equality holds in Kraft's inequality

$$\sum_{u \in W} \alpha^{-|u|} = 1. \tag{3.2}$$

Applying Kraft's inequality to the prefix set $\Phi(W)$ then yields

$$\sum_{u \in W} 2^{-|\Phi(u)|} = \sum_{v \in \Phi(W)} 2^{-|v|} \le 1. \tag{3.3}$$

Since the summation on the left side of (3.3) is upper-bounded by the summation on the left side of (3.2), there must exist $u \in W$ such that $2^{-|\Phi(u)|} \le \alpha^{-|u|}$. This inequality is equivalent to inequality (3.1).

Proof of Theorem 1: In view of Facts 3.1 and 3.2, Theorem 1 will automatically hold for all sequential codes in Σ once we prove Theorem 1 for all sequential codes in Σ'. Thus to prove Theorem 1, we fix $\sigma = \{(W_n, \Psi_n, \Phi_n)\}$, an arbitrary code in Σ', and show that the conclusions of Theorem 1 follow for this code. First, we argue that there is a total recursive function $Q: N \to A^*$ such that

$$Q(n) \in W_n$$

and

$$\left|\Phi_n\big(Q(n)\big)\right| \ge |Q(n)| \log \alpha, \quad n \in N. \tag{3.4}$$

To see this, we shall envision a computer which halts and prints out the desired $Q(n)$ in response to input n, for an arbitrary n. Let $u_1, u_2, u_3, \cdots$ be the list of all A-strings formed by listing the strings of length one, followed by the strings of length two, etc., with strings of the same length listed in lexicographical order. There is some Turing machine T_1 such that if we apply an input of the form (n, initial segment of u_i), T_1 will halt and print either "initial segment of u_i is in W_n" or "initial segment of u_i is not in W_n." There is a second Turing machine T_2 which will halt and print Φ_n (initial segment of u_i) in response to an input of form (n, initial segment of u_i), provided the initial segment of u_i is in W_n. Run the following program on a computer:

1. Read in a value of n.
2. Set $i = 1$.
3. Set initial segment of u_i = first symbol of u_i.
4. Run T_1 with input (n, initial segment of u_i). If output is "initial segment of u_i is in W_n," go to instruction **5**; otherwise, go to instruction **6**.
5. Run T_2 with input (n, initial segment of u_i) to compute Φ_n(initial segment of u_i). If $|\Phi_n$(initial segment of $u_i)| \ge (\log \alpha)|$ initial segment of $u_i|$, go to instruction **7**; otherwise, go to instruction **6**.
6. If "initial segment of u_i" is a proper prefix of u_i, augment "initial segment of u_i" by one symbol and go to instruction **4**; otherwise, increase i by one and go to instruction **3**.
7. Set $Q(n)$ = initial segment of u_i and halt.

By Lemma 1, the computation will halt for every n; appealing to the closure property of the class of recursive functions, one concludes that the above program defines a total recursive function $Q: N \to A^*$ satisfying (3.4). Again, from

the closure property, one obtains the total recursive function $R: N \to N$ in which

$$R(1) = 1; \quad R(n+1) = \Psi_{R(n)}\Big(Q\Big(R(n)\Big)\Big), \quad n \geq 1.$$

Let x' be the individual sequence

$$x' = Q\Big(R(1)\Big) * Q\Big(R(2)\Big) * Q\Big(R(3)\Big) \cdots .$$

The sequence x' is recursive, and conclusion i) of Theorem 1 holds. For each $n \in N$ and $y \in W_n$, let $H(n, y)$ denote the binary string

$$H(n, y) = \begin{cases} 0, & y = Q(n) \\ 1 * \Phi_n(y), & \text{otherwise.} \end{cases}$$

If we regard $H(n, y)$ as a function of n, y, it is a recursive function. Furthermore, for each $n \in N$, the function $\Phi'_n: W_n \to \{0, 1\}^*$ defined by

$$\Phi'_n(y) \triangleq H(n, y), \quad y \in W_n$$

is one-to-one, and $\Phi'_n(W_n)$ is a prefix set. Consequently, $\sigma' = \{(W_n, \Psi_n, \Phi'_n)\}$ is a sequential code belonging to the class Σ. Due to Property 3.1, it is a simple matter to verify conclusions ii) and iii) of Theorem 1 for the code σ'.

IV. Kolmogorov Complexity and the OPTA Function

Our task in this section is to show that the OPTA function does not coincide with the Kolmogorov complexity rate function. We start with the definition of Kolmogorov complexity. Informally speaking, the Kolmogorov complexity of an A-string is the length of the shortest program in bits, which, when run on a universal computer, will cause the computer to generate the given A-string as output. To formalize this notion, we need to fix a recursive function $U: \{0, 1\}^* \to A^*$ which has the following property: If $f: \{0, 1\}^* \to A^*$ is any recursive function, then there exists $p \in \{0, 1\}^*$ such that

i) $U(p * y)$ is defined whenever $f(y)$ is defined; and
ii) $U(p * y) = f(y)$, for every y in the domain of f.

It is known that there exist many recursive functions U satisfying this property; any such function shall be called a *universal* recursive function. If y is an A-string, the Kolmogorov complexity of y [4], [6], [9], [10], [14], is the integer $\boldsymbol{K}_U(y)$ defined as follows:

$$\boldsymbol{K}_U(y) \triangleq \min\{|p|: U(p) = y\}.$$

We then define the Kolmogorov complexity rate function to be the function $\overline{\boldsymbol{K}}: A^\infty \to [0, \infty)$ in which

$$\overline{\boldsymbol{K}}(x) \triangleq \limsup_{n\to\infty} \frac{\boldsymbol{K}_U(x^n)}{n}$$

for any individual sequence x. The Kolmogorov complexity rate function does not depend upon the choice of universal recursive function U. This follows from the invariance principle [9, ch. 2] which states that for any two universal recursive functions U_1, U_2

$$\sup\{|\boldsymbol{K}_{U_1}(x) - \boldsymbol{K}_{U_2}(x)|: x \in A^*\} < \infty.$$

The following result shows that the Kolmogorov complexity rate function provides a lower bound to the OPTA function.

Theorem 2: For any individual sequence x, $\rho(x) \geq \overline{\boldsymbol{K}}(x)$.

Proof: Fix an arbitrary individual sequence x, and an arbitrary sequential code $\sigma \in \Sigma$. The theorem is established if we can show that $\rho(x|\sigma) \geq \overline{\boldsymbol{K}}(x)$. By Lemma I.1 of Appendix I, there exists a sequence $\{\epsilon_n\}$ converging to zero as $n \to \infty$ and a one-to-one total recursive function $f: A^* \to \{0, 1\}^*$ such that

$$|f(x^n)| \leq L(\sigma, x^n) + n\epsilon_n, \quad n \in N.$$

One can show (Lemma II.3, Appendix II) that there is a recursive function $g: \{0, 1\}^* \to A^*$ such that

$$g\Big(f(y)\Big) = y, \quad \text{for any } y \in A^*.$$

Fix $p \in \{0, 1\}^*$ such that $U(p * y) = g(y)$ whenever $g(y)$ is defined. Then $U\Big(p * f(x^n)\Big) = g\Big(f(x^n)\Big) = x^n$ and so

$$\boldsymbol{K}_U(x^n) \leq |p| + L(\sigma, x^n) + n\epsilon_n, \quad n \in N.$$

Dividing by n and letting $n \to \infty$, we obtain the desired conclusion.

We are now ready to state the main result of this paper:

Theorem 3: There exists an individual sequence x such that $\overline{\boldsymbol{K}}(x) = 0$ and $\rho(x) = \log \alpha$.

Remark: There is an interesting result of Ya. M. Barzdin and N. V. Petri ([14, Theorem 2.5]) that is related to our Theorem 3. This result concerns binary individual sequences, i.e., individual sequences in which the underlying alphabet is $A = \{0, 1\}$. A binary individual sequence x is said to be recursively enumerable if there is a total recursive function $g: N \to N$ such that $g(N) = \{n: x^n \text{ ends in } 1\}$. Let $\mathcal{F}$ be the family of all total recursive functions $f: A^* \to N$ such that $f(y) \geq K_U(y)$ for every A-string y. Then, the Barzdin–Petri result states that there exists a recursively enumerable binary individual sequence x such that $\inf_n n^{-1} f(x^n) > 0$ for any $f \in \mathcal{F}$. Since any recursively enumerable binary individual sequence x satisfies $K_U(x^n) = O(\log n)$ ([2, Theorem 1]), the Barzdin–Petri result guarantees the existence of a binary individual sequence x for which $\overline{K}(x) = 0$ and for which $\rho(x|\sigma) > 0$ for any $\sigma \in \Sigma$. Note that one cannot deduce $\rho(x) > 0$ from the fact that $\rho(x|\sigma) > 0$ for any $\sigma \in \Sigma$. We will go further and show the existence of an α-ary individual sequence x for which $\overline{K}(x) = 0$ and $\rho(x|\sigma) \geq \log \alpha$ for any $\sigma \in \Sigma$.

Proof of Theorem 3: Choose a sequence of sequential codes $\{\sigma_i: i \in N\}$ from Σ' such that every member of Σ' appears infinitely often in $\{\sigma_i\}$. (Since Σ is countable, it is possible to select such a sequence $\{\sigma_i\}$.) For each i, let π_i denote the parsing rule underlying σ_i. For each i, by Lemma II.4 of Appendix II, there exists a total recursive function $Q_i: A^* \to A^*$ such that all of the following four statements are true:

1) y is a proper prefix of $Q_i(y)$, $y \in A^*$.
2) The π_i-parsing of $Q_i(y)$ is complete, $y \in A^*$.
3) If $\{u_1, u_2, \cdots, u_t\}$ is the π_i-parsing of $y \in A^*$, then there exists $u \in A^*$ such that $\{u_1, u_2, \cdots, u_t, u\}$ is the π_i-parsing of $Q_i(y)$.

4) If $\{v_t: t \in N\}$ is a sequence of iterates under Q_i (meaning $v_{t+1} = Q_i(v_t)$, $t \in N$), then

$$\liminf_{t\to\infty} L(\sigma_i, v_t)/|v_t| \geq \log \alpha.$$

For each i, it follows from the fact that Q_i is recursive (see Lemma II.2 of Appendix II) that there must be a positive integer D_i such that

$$\boldsymbol{K}_U\Big(Q_i(y)\Big) \leq \boldsymbol{K}_U(y) + D_i, \quad y \in A^*. \tag{4.1}$$

Choose an increasing sequence of positive integers $\{n_i: i \in N\}$ and A-strings $\{v_{i,j}: i \in N, 1 \leq j \leq n_i\}$ such that all of the following are true:

$$n_i \geq [iD_i]^2 + [(i+1)D_{i+1}]^4, \quad i \in N. \tag{4.2}$$

$$\begin{aligned} v_{i,j} &= Q_i(v_{i,j-1}), \quad i \in N, 2 \leq j \leq n_i. \\ v_{i,1} &= Q_i(v_{i-1,n_{i-1}}), \quad i \geq 2; \ \ v_{1,1} = Q_1(\lambda_A). \\ L(\sigma_i, v_{i,n_i})/|v_{i,n_i}| &\geq (1 - i^{-1}) \log \alpha, \quad i \in N. \end{aligned} \tag{4.3}$$

Let x be the individual sequence such that $v_{i,1}$ is a prefix of x for every i. Because of (4.3), Fact 3.3, and the fact that every code in Σ' appears in $\{\sigma_i\}$ infinitely many times, we must have $\rho(x) = \log \alpha$.

To complete the proof of Theorem 3, we show that $\overline{\boldsymbol{K}}(x) = 0$. Define

$$v_{i,0} \triangleq v_{i-1,n_{i-1}}, \quad i \geq 2; \ \ v_{1,0} \triangleq \lambda_A.$$

For $i \in N$ and $1 \leq j \leq n_i$, define $u_{i,j}$ to be $v_{i,j}$ with the initial segment $v_{i,j-1}$ removed. Define

$$u^i \triangleq u_{i,1} * u_{i,2} * \cdots * u_{i,n_i}, \quad i \in N.$$

For each $m > |u^1|$, let $i = i_m \geq 2$ be the integer such that $u^1 * u^2 * \cdots * u^{i-1}$ is a prefix of x^m and x^m is a prefix of $u^1 * u^2 * \cdots * u^i$. Then, choose $j = j_m$ to be the positive integer such that x^m is a prefix of

$$u^1 * u^2 * \cdots * u^{i-1} * u_{i,1} * u_{i,2} * \cdots * u_{i,j} = v_{i,j}$$

and $v_{i,j-1}$ is a prefix of x^m. By Lemma II.1 of Appendix II, there exists $\beta \in N$ such that

$$\boldsymbol{K}_U(x^m) \leq \beta \log(m+1) + \boldsymbol{K}_U(v_{i_m,j_m}), \quad m > |u^1|.$$

Consequently, $\overline{\boldsymbol{K}}(x) = 0$ will follow if we can show that

$$\lim_{i\to\infty} \max_{1\leq j\leq n_i} \frac{\boldsymbol{K}_U(v_{i,j})}{|v_{i,j-1}|} = 0. \tag{4.4}$$

By construction, $v_{i,j}$ is obtained from $v_{1,0} = \lambda_A$ via n_1 applications of Q_1, n_2 applications of $Q_2, \cdots, n_{i-1}$ applications of Q_{i-1}, and finally, j applications of Q_i. Making use of (4.1), we then have

$$\begin{aligned} \boldsymbol{K}_U(v_{i,j}) \leq\ & n_1D_1 + n_2D_2 + \cdots + n_{i-1}D_{i-1} + jD_i \\ & + \boldsymbol{K}_U(\lambda_A), \quad i \geq 2, 1 \leq j \leq n_i. \end{aligned}$$

By Property 3.1,

$$|u^i| \geq 1 + \sqrt{2} + \sqrt{3} + \cdots + \sqrt{n_i}, \quad i \in N.$$

Since

$$\sum_{s=1}^{r} \sqrt{s} \geq \int_0^r \sqrt{z}\, dz = \tfrac{2}{3} r^{3/2}, \quad r \in N$$

we can conclude that

$$|u^i| \geq \tfrac{2}{3} n_i^{3/2}, \quad i \in N \tag{4.5}$$

whence

$$n_i \leq 2n_i^{-1/2}|u^i|, \quad i \in N.$$

It follows from this and (4.2) that

$$\begin{aligned} & n_1D_1 + n_2D_2 + \cdots + n_{i-1}D_{i-1} \\ & \quad \leq 2[D_1 n_1^{-1/2}|u^1| + \cdots + D_{i-1}n_{i-1}^{-1/2}|u^{i-1}|] \\ & \quad \leq 2[|u^1| + |u^2|/2 + \cdots + |u^{i-1}|/(i-1)], \quad i \geq 2. \end{aligned}$$

Fix j, i where $1 \leq j \leq n_i$ and $i \geq 2$. If $j \leq \sqrt{|v_{i,j-1}|}$

$$\frac{jD_i}{|v_{i,j-1}|} \leq \frac{D_i}{\sqrt{|v_{i,j-1}|}} \leq \frac{D_i}{\sqrt{n_{i-1}}} \leq \frac{1}{i}.$$

Now suppose $j > \sqrt{|v_{i,j-1}|}$. Then $j \geq 2$ and so $j - 1 \geq j/2$. Arguing as in the proof of (4.5), we have

$$|v_{i,j-1}| \geq (2/3)(j-1)^{3/2} \geq (2/3)\,(j/2)^{3/2}.$$

One then deduces that

$$\frac{jD_i}{|v_{i,j-1}|} \leq \frac{5D_i}{\sqrt{j}} \leq \frac{5D_i}{|v_{i,j-1}|^{1/4}} \leq \frac{5D_i}{n_{i-1}^{1/4}} \leq \frac{5}{i}.$$

We conclude that for $i \geq 2$

$$\begin{aligned} \max_{1\leq j\leq n_i} \frac{\boldsymbol{K}_U(v_{i,j})}{|v_{i,j-1}|} \leq\ & 2\left[\frac{|u^1| + |u^2|/2 + \cdots + |u^{i-1}|/(i-1)}{|u^1| + |u^2| + \cdots + |u^{i-1}|}\right] \\ & + \frac{5}{i} + \frac{K_U(\lambda_A)}{|u^{i-1}|}. \end{aligned}$$

It is easy to see that the right-hand side of the above inequality converges to zero as $i \to \infty$. We see now that (4.4) is true, completing the proof of Theorem 3.

V. Conclusions

The answers to Questions 1 and 2 obtained in this paper lead one to the following conclusions. First of all, one should not waste time trying to find a "best" sequential code, as any sequential code from the class Σ will have an inherent deficiency in that it will poorly compress certain individual sequences. Secondly, the Kolmogorov complexity rate function is not the right notion of complexity vis-a-vis the OPTA function in lossless individual sequence compression in that it coincides with the OPTA function only on a proper subset of the set of all individual sequences.

Appendix I

This appendix is devoted to the proof of Facts 1.1–1.3 that were stated in Section I.

Lemma I.1: Let $\sigma \in \Sigma$ be arbitrary. Then there exists a one-to-one total recursive function f from A^* onto a prefix subset of $\{0, 1\}^*$, and a sequence $\{\epsilon_n : n \in N\}$ converging to zero as $n \to \infty$ such that

$$|f(y)| \le L(\sigma, y) + n\epsilon_n, \quad y \in A^n, n \in N. \tag{A1}$$

Proof: Let $R: N \to \{0, 1\}^*$ be the mapping in which, for each $k \in N$, the binary string $R(k)$ is formed in the following way: First, form the expansion of k to base two; then, form $R(k)$ by writing every digit in the expansion twice, followed by the suffix "01." For example, the base two expansion of the integer 9 is 1001. Repeating each digit and appending the suffix "01" yields $R(9) = 1100001101$. For later use, the reader can easily check that $|R(k)| \le 6 + 2\log k$, $k \in N$. It is easy to define a total recursive function $Q: A^* \to \{0, 1\}^*$ in which

1) $|Q(v)| = \lceil |v| \log \alpha \rceil$, $v \in A^*$.
2) $Q(v_1) \ne Q(v_2)$ if v_1 and v_2 have the same length and $v_1 \ne v_2$.

We fix $\sigma = \{(W_n, \Psi_n, \Phi_n)\}$, an arbitrary sequential code from the class Σ. Let $H: N \times A^* \to \{0, 1\}^*$ be a recursive function such that

$$H(n, y) = \Phi_n(y), \quad n \in N, y \in W_n.$$

If $y \in A^*$, let $t = t(y)$ denote the number of phrases in the π_σ-parsing of y, let $\{u_1, u_2, \cdots, u_t\}$ denote the π_σ-parsing of y, let

$$j = j(y) = |u_1| + |u_2| + \cdots + |u_t|$$

and let $\hat{y}$ denote the suffix of y consisting of the final $|y| - j(y)$ α-ary symbols in y. Define $f: A^* \to \{0, 1\}^*$ by

$$f(y) \triangleq \begin{cases} R(|y|+1) * R\Big(j(y)+1\Big) * Q(y), & \text{if } t(y) = 0 \\ R(|y|+1) * R\Big(j(y)+1\Big) * Q(\hat{y}) \\ \quad * H(i_1, u_1) * \cdots * H(i_t, u_t) & \text{otherwise} \end{cases}$$

where $\{i_1, i_2, \cdots, i_t\}$ are the parser states

$$i_1 = 1, \quad i_k = \Psi_{i_{k-1}}(u_{k-1}), \quad 2 \le k \le t.$$

Then f is a one-to-one mapping, and $f(A^*)$ is a prefix set. That f is recursive follows from the fact that the functions R, Q, and H are recursive, as well as the fact that each of the five entities t, j, $\hat{y}$, $\{u_1, u_2, \cdots, u_t\}$, and $\{i_1, i_2, \cdots, i_t\}$, regarded as a function of $y \in A^*$, is a recursive function. From the definition of f, we have the bound

$$|f(y)| \le 12 + 4\log(|y| + 1) + L(\sigma, y) + \left\lceil \Big(|y| - j(y)\Big) \log \alpha \right\rceil, \quad y \in A^*. \tag{A2}$$

From Property 2.1, we have

$$\lim_{|y| \to \infty} \frac{|y| - j(y)}{|y|} = 0. \tag{A3}$$

The relation (A1) is now evident from (A2) and (A3).

The following result is an immediate corollary of Lemma I.1.

Corollary I.1: Let $\sigma \in \Sigma$ be arbitrary. Then there is a sequence $\{f_n : n \in N\}$ in which

i) The function f_n is a one-to-one map of A^n onto a prefix subset of $\{0, 1\}^*$, $n \in N$.
ii) For any $x \in A^\infty$,

$$\rho(x|\sigma) \ge \limsup_{n \to \infty} \frac{|f_n(x^n)|}{n}.$$

Proof of Fact 1.1: Let μ be a probability measure on A^∞ stationary and ergodic with respect to the one-sided shift transformation. Let $H(\mu)$ denote the entropy rate of μ (in bits per α-ary symbol). Let $f_n: A^n \to \{0, 1\}^*$ be a one-to-one map such that $f_n(A^n)$ is a prefix set, $n \in N$. It is well known [1, Theorem 3.1], [5] that

$$\mu\left\{x \in A^\infty : \liminf_{n \to \infty} \frac{|f_n(x^n)|}{n} \ge H(\mu)\right\} = 1. \tag{A4}$$

Taking the infimum over all sequences $\{f_n\}$ then yields, in view of the Corollary to Lemma I.1,

$$\mu\{x \in A^\infty : \rho(x) \ge H(\mu)\} = 1. \tag{A5}$$

But, Ziv and Lempel [13] showed that

$$\mu\{x \in A^\infty : \rho(x|\sigma_{LZ}) \le H(\mu)\} = 1. \tag{A6}$$

Combining these last two equalities leads one to the assertion that $\mu(\Omega_1) = 1$. That this equality holds also for μ stationary but not ergodic is a consequence of the ergodic decomposition theorem.

Proof of Fact 1.2: Let μ be as in the proof of Fact 1.1. The assertion of Fact 1.2 will follow if we can show that $\mu(\Omega_2) = 1$. It is clear from the proof of Lemma I.1 that we may assign to each $p \in \{0, 1\}^*$ a string $s(p) \in \{0, 1\}^*$ of length $o(|p|)$ such that $\{s(p) * p : p \in \{0, 1\}^*\}$ forms a prefix set. For $n \in N$, define $f_n: A^n \to \{0, 1\}^*$ by

$$f_n(y) \triangleq s(p) * p, \text{ for some } p \in \{0, 1\}^* \text{ such that } U(p) = y \text{ and } |p| = \boldsymbol{K}_U(y), \; y \in A^n.$$

Note that f_n is a one-to-one mapping and that $f_n(A^n)$ is a prefix set, $n \in N$. Consequently, (A4) holds. Also note that

$$\overline{\boldsymbol{K}}(x) = \limsup_{n \to \infty} \frac{|f_n(x^n)|}{n}, \quad x \in A^\infty. \tag{A7}$$

We conclude that

$$\mu\{x \in A^\infty : \overline{\boldsymbol{K}}(x) \ge H(\mu)\} = 1. \tag{A8}$$

(Alternatively, one can deduce (A8) from the stronger statement that $\overline{\boldsymbol{K}}(x) = H(\mu)$ for μ-almost all x, a result which may be found in [8], [14].) Combining (A5) with (A6), we have

$$\mu\{x \in A^\infty : \rho(x) = H(\mu)\} = 1. \tag{A9}$$

Combining (A8), (A9), and Theorem 2, we obtain $\mu(\Omega_2) = 1$.

Proof of Fact 1.3: Let μ be a computable probability measure on A^∞. In this proof, if λ is a probability measure on A^∞ and $y \in A^*$, we define

$$\lambda(y) = \lambda\{x \in A^\infty : x \text{ starts with } y\}.$$

From the proof of Fact 1.2, there exists for each $n \in N$ a one-to-one mapping f_n from A^n onto a prefix subset of $\{0, 1\}^*$ such that (A7) holds. Then, applying Kraft's inequality

$$\mu\left\{x \in A^\infty : \frac{|f_n(x^n)| + \log \mu(x^n)}{n} \le -\epsilon\right\} \le 2^{-n\epsilon}, \quad \epsilon > 0. \tag{A10}$$

(Note that the logarithm in (A10) makes sense because $\mu\{x \in A^\infty : \mu(x^n) > 0 \text{ all } n\} = 1$.) Applying the Borel–Cantelli lemma to (A10), we obtain

$$\mu\left\{x \in A^\infty : \limsup_{n\to\infty} \frac{|f_n(x^n)|}{n} \ge \limsup_{n\to\infty} \frac{-\log \mu(x^n)}{n}\right\} = 1$$

whence, appealing to (A7), we conclude that

$$\mu\left\{x \in A^\infty : \overline{\boldsymbol{K}}(x) \ge \limsup_{n\to\infty} \frac{-\log \mu(x^n)}{n}\right\} = 1. \tag{A11}$$

Fix a computable probability measure λ on A^∞ such that $\lambda(y) > 0$ for any $y \in A^*$. Let μ^* be the probability measure defined by

$$\mu^* \triangleq \tfrac{1}{2}(\mu + \lambda).$$

Let m be any positive integer. Since μ^* is computable, there exists (as shown in [11]) a sequential code $\sigma_m = \{(W_n, \Psi_n, \Phi_n)\}$ from Σ in which

i) The parsing rule π_{σ_m} is the m-block parsing rule; and
ii) For any individual sequence x, if $\{u_t : t \in N\}$ is the π_{σ_m}-parsing of x and $\{i_t : t \in N\}$ is given by (2.1), then

$$|\Phi_{i_t}(u_t)| \le 2 - \log \frac{\mu^*(u_1 * \cdots * u_{t-1} * u_t)}{\mu^*(u_1 * \cdots * u_{t-1})}, \quad t \ge 2.$$

From i) and ii) it follows that

$$\rho(x|\sigma_m) \le \frac{2}{m} + \limsup_{n\to\infty} -\frac{\log \mu^*(x^n)}{n}, \quad m \in N,\ x \in A^\infty$$

from which we conclude that

$$\rho(x) \le \limsup_{n\to\infty} -\frac{\log \mu^*(x^n)}{n}, \quad x \in A^\infty. \tag{A12}$$

Since $\mu^*(x^n) \ge \mu(x^n)/2$, we see that

$$\mu\left\{x \in A^\infty : \limsup_{n\to\infty} -\frac{\log \mu^*(x^n)}{n} \le \limsup_{n\to\infty} -\frac{\log \mu(x^n)}{n}\right\} = 1.$$

This fact, together with (A11) and (A12), allows us to conclude that

$$\mu\{x \in A^\infty : \overline{\boldsymbol{K}}(x) \ge \rho(x)\} = 1.$$

This relation, coupled with Theorem 2, allows us to conclude that $\mu(\Omega_2) = 1$.

APPENDIX II

The purpose of this appendix is to prove several lemmas which are instrumental in the proofs of Theorems 1–3.

Lemma II.1: There exists a constant β which depends only on U such that

$$\boldsymbol{K}_U(v_1) \le \beta + 2\log(|v_1| + 1) + \boldsymbol{K}_U(v_2) \tag{A13}$$

whenever v_1, v_2 are strings in A^* for which v_1 is a prefix of v_2.

Proof: Let $R: N \to \{0, 1\}^*$ be the total recursive function defined in the proof of Lemma I.1. Since $R(N)$ is a prefix set, we can define a function $T: \{0, 1\}^* \to A^*$ in the following way: i) the domain of T is equal to

$$\{R(k) * y : k \in N,\ y \in \{0, 1\}^*,\ |U(y)| \ge k - 1\}$$

and ii)

$$T(R(k) * y) = U(y)^{k-1}, \quad \text{if } k \in N,\ y \in \{0, 1\}^*, \text{ and } |U(y)| \ge k - 1.$$

Since R and U are both recursive functions, it follows that T is a recursive function. Let $p \in \{0, 1\}^*$ be a string such that

$$T(u) = U(p * u), \quad u \in \text{domain of } T.$$

We verify (A13) with $\beta = |p| + 6$. Choose any pair of strings v_1, v_2 from A^* with v_1 a prefix of v_2. Pick $y_2 \in \{0, 1\}^*$ such that $|y_2| = \boldsymbol{K}_U(v_2)$ and $U(y_2) = v_2$. Let $\tau = |v_1|$. Then $U\Big(p * R(\tau + 1) * y_2\Big) = v_1$ and so

$$\begin{aligned}\boldsymbol{K}_U(v_1) &\le |p * R(\tau + 1) * y_2| \\ &\le |p| + 6 + 2\log(|v_1| + 1) + \boldsymbol{K}_U(v_2).\end{aligned}$$

Lemma II.2: Let $F: A^* \to A^*$ be a total recursive function. Then there exists a positive integer D such that

$$\boldsymbol{K}_U\Big(F(y)\Big) \le \boldsymbol{K}_U(y) + D, \quad y \in A^*. \tag{A14}$$

Proof: Let $V: \{0, 1\}^* \to A^*$ be the mapping whose domain is the same as that of U, and $V(z) = F\big(U(z)\big)$ for z in the domain of U. Since V is a recursive function (the composition of two recursive functions is a recursive function), there must exist $p \in \{0, 1\}^*$ such that $U(p * z) = V(z)$ for every z in the domain of V. We show (A14) holds with $D = |p|$. Fix $y \in A^*$ and choose $z \in \{0, 1\}^*$ such that $U(z) = y$ and $|z| = \boldsymbol{K}_U(y)$. Then $U(p * z) = F(y)$ and so

$$\boldsymbol{K}_U\Big(F(y)\Big) \le |p * z| = |p| + \boldsymbol{K}_U(y).$$

Lemma II.3: Let $f: A^* \to \{0, 1\}^*$ be a one-to-one total recursive function. Then the function $g: \{0, 1\}^* \to A^*$ defined by

$$g(z) \triangleq \begin{cases} \text{undefined}, & \text{if } z \notin f(A^*) \\ \text{the unique } y \in A^* \text{ such that } f(y) = z, & \text{otherwise} \end{cases}$$

is recursive.

Proof: We impose a total order $<$ on A^* as follows: $v_1 < v_2$ if $|v_1| < |v_2|$; and $v_1 < v_2$ if $|v_1| = |v_2|$ and v_1 precedes v_2 in the lexicographical ordering. Let $Q: A^* \times \{0, 1\}^* \to N$ be the total recursive function defined by

$$Q(y, z) \triangleq \begin{cases} 1, & \text{if } z = f(y) \\ 2, & \text{otherwise.} \end{cases}$$

Let $q: \{0, 1\}^* \to A^*$ be the function defined by

$$q(z) \triangleq \begin{cases} \text{undefined}, & \text{if there is no } y \text{ such that } Q(y, z) = 1 \\ \min\{y \in A^*: Q(y, z) = 1\}, & \text{otherwise.} \end{cases}$$

By the closure property of the class of recursive functions, the function q is recursive. It is easy to check that the functions g and q coincide.

Lemma II.4: Let $\sigma \in \Sigma$ be arbitrary. Then there exists a total recursive function $Q: A^* \to A^*$ such that all of the following are true:

i) The string y is a proper prefix of $Q(y)$, $y \in A^*$.
ii) The π_σ-parsing of $Q(y)$ is complete, $y \in A^*$.
iii) If $\{u_1, u_2, \cdots, u_t\}$ is the π_σ-parsing of $y \in A^*$, then there exists $u \in A^*$ such that $\{u_1, u_2, \cdots, u_t, u\}$ is the π_σ-parsing of $Q(y)$.
iv) If $\{v_t: t \in N\}$ is a sequence of iterates under Q, then

$$\liminf_{t \to \infty} L(\sigma, v_t)/|v_t| \geq \log \alpha.$$

Proof: Let $\sigma = \{(W_n, \Psi_n, \Phi_n)\}$. Let F, G, H be the recursive functions satisfying Properties 2.2–2.4, respectively. Suppose $y \in A^*$. Let $\{u_1, u_2, \cdots, u_t\}$ denote the π_σ-parsing of y. Define $\tilde{y} = u_1 * u_2 * \cdots * u_t$. If $\tilde{y} = \lambda_A$, define $i(y) = 1$; otherwise, define $i(y) = i_{t+1}$, where $i_1, i_2, \cdots, i_t, i_{t+1}$ are the integers

$$i_1 = 1;\ i_k = G(i_{k-1}, u_{k-1}), \quad 1 \leq k \leq t + 1.$$

(In other words, $i(y)$ is the final parser state in the parsing of y.) Define $\hat{y}$ to be the suffix of y such that $y = \tilde{y} * \hat{y}$. Totally order A^* as in the proof of Lemma II.3. Define $Q: A^* \to A^*$ by

$$Q(y) \triangleq \tilde{y} * u$$

where $u \in A^*$ is obtained as follows:

$$u \triangleq \begin{cases} \min\{v \in A^*: F[i(y), v] = 1 \\ \quad \text{and } |H[i(y), v]| \geq |v| \log \alpha\}, & \text{if } \tilde{y} = y \\ \min\{v \in A^*: F[i(y), v] = 1 \\ \quad \text{and } v \text{ starts with } \hat{y}\}, & \text{otherwise.} \end{cases}$$

Since the functions F, G, H are recursive and the functions $y \to \tilde{y}$, $y \to \hat{y}$, and $y \to i(y)$ are total recursive, it follows that Q is a total recursive function. Conclusions i)–iv) are easily verified.

Proof of Fact 3.2: Let $\sigma = \{(W_n, \Psi_n, \Phi_n)\} \in \Sigma$ be arbitrary. We shall construct $\sigma' \in \Sigma'$ subordinate to σ. Let T be the one-sided shift transformation on A^∞. We define a transformation R on the set $\Omega = A^\infty \times N^3$ as follows. If $(x, n, j, s) \in \Omega$, let $R(x, n, j, s) = (x', n', j', s')$, where

$x' = T^{|v_n|}(x)$, v_n = unique prefix of x in W_n;
$n' = \Psi_n(v_n)$;
$j' = j + 1$;
$s' = s + |v_n|$, if $s + |v_n| \leq \sqrt{j}$; and $s' = 1$, otherwise.

Fix an arbitrary pair $(n, j) \in N^2$. We define a mapping $t_{n,j}: A^\infty \to N$ as follows. Let $x \in A^\infty$. Let

$$\left(x^{(m)}, n^{(m)}, j^{(m)}, s^{(m)}\right), \quad m = 1, 2, \cdots$$

be the sequence in Ω in which

$$\left(x^{(1)}, n^{(1)}, j^{(1)}, s^{(1)}\right) = R(x, n, j, 1)$$

$$\left(x^{(m+1)}, n^{(m+1)}, j^{(m+1)}, s^{(m+1)}\right) = R\left(x^{(m)}, n^{(m)}, j^{(m)}, s^{(m)}\right), \quad m \geq 1.$$

Then

$$t_{n,j}(x) \triangleq \min\{m \in N: s^{(m)} = 1\}.$$

Let $W_{n,j}$ be the finite subset of A^* consisting of all strings $u_1 * u_2 * \cdots * u_t$, $t \in N$, for which there exists some $x \in A^\infty$ such that $t_{n,j}(x) = t$ and $u_1, u_2, \cdots u_t$ are the first t phrases in the (π_σ, n)-parsing of x. From the definition of the mapping $t_{n,j}$, one can see that $W_{n,j}$ is a complete prefix subset of A^*. Note that if $y \in W_{n,j}$, then y can be decomposed as

$$y = u_1 * u_2 * \cdots * u_t \tag{A15}$$

where $\{u_1, u_2, \cdots, u_t\}$ is the (π_σ, n)-parsing of y. We now define the mappings $\Psi_{n,j}: W_{n,j} \to N^2$ and $\Phi_{n,j}: W_{n,j} \to A^*$ for which, with $y \in W_{n,j}$ decomposed as in (A15)

$$\Psi_{n,j}(y) \triangleq (i_{t+1}, t + j)$$

and

$$\Phi_{n,j}(y) \triangleq \Phi_{i_1}(u_1) * \Phi_{i_2}(u_2) * \cdots * \Phi_{i_t}(u_t)$$

where

$$i_1 = n,\ i_k = \Psi_{i_{k-1}}(u_{k-1}), \quad 2 \leq k \leq t + 1.$$

Fix any one-to-one map τ of N onto N^2 such that $\tau(1) = (1, 1)$ and both τ and τ^{-1} are total recursive. Let $\sigma' = \{(W'_m, \Psi'_m, \Phi'_m): m \in N\}$ be the sequential code in which

1) W'_m is the complete prefix subset of A^* such that $W'_m = W_{n,j}$, where $(n, j) = \tau(m)$.
2) Ψ'_m is the mapping from W'_m to N such that $\tau(\Psi'_m(y)) = \Psi_{n,j}(y)$ for each $y \in W'_m$, where $(n, j) = \tau(m)$.
3) Φ'_m is the mapping from W'_m to $\{0, 1\}^*$ such that $\Phi'_m(y) = \Phi_{n,j}(y)$ for each $y \in W'_m$, where $(n, j) = \tau(m)$.

We want to show that σ' belongs to the class Σ. Properties 2.2–2.4 are valid for σ' because they are valid for σ and because of the way in which σ' was defined using σ. All that remains to be shown is the validity of Property 2.1 for σ', which means we must show that

$$\omega_{\tilde{\pi}}(Q) \to 0 \quad \text{as} \quad Q \to \infty \tag{A16}$$

where $\tilde{\pi}$ denotes the parsing rule underlying σ'. We shall establish the bound

$$\omega_{\tilde{\pi}}(Q) \le \frac{3}{\sqrt{Q+1}-1} + 9\omega_{\pi_\sigma}(Q), \quad Q \in N, \tag{A17}$$

from which (A16) is apparent. Fix an integer $Q \in N$. Let $q \ge Q$, and $x \in A^\infty$. Let $\{u_i : i \in N\}$ denote the π_σ-parsing of x and let $\{\tilde{u}_i : i \in N\}$ denote the $\tilde{\pi}$-parsing of x. We will show that

$$\frac{|\tilde{u}_{q+1}|}{|\tilde{u}_1| + |\tilde{u}_2| + \cdots + |\tilde{u}_q|} \le \frac{3}{\sqrt{Q+1}-1} + 9\omega_{\pi_\sigma}(Q) \tag{A18}$$

from which the bound (A17) follows. There exist positive integers $s \le t$ such that the following three relations hold:

$$\tilde{u}_{q+1} = u_s * u_{s+1} * \cdots * u_t.$$

$$|\tilde{u}_{q+1}| > \sqrt{t}. \tag{A19}$$

$$|u_s| + |u_{s+1}| + \cdots + |u_{t-1}| \le \sqrt{t-1}, \quad \text{if } s < t. \tag{A20}$$

We deduce the relationship

$$\frac{\sqrt{t}}{s-1} \le \frac{\sqrt{2}}{\sqrt{s}-1}. \tag{A21}$$

To establish this, one first observes that

$$s \ge t - \sqrt{t-1}. \tag{A22}$$

(If $s < t$, this follows from (A20); if $s = t$, this is trivial.) Solving the inequality (A22) for t yields (using the quadratic formula)

$$t \le s + \tfrac{1}{2} + \tfrac{1}{2}\sqrt{4s-3}. \tag{A23}$$

Upper bounding the right side of (A23) furnishes us with the bounds

$$t \le s + \tfrac{1}{2} + \sqrt{s} \le (\sqrt{s} + \tfrac{1}{2})^2 + \tfrac{1}{4} \le 2(\sqrt{s}+1)^2$$

from which (A21) is evident. From (A19) and (A20), it can be seen that one of the following two statements must hold:

4) $|\tilde{u}_{q+1}| \le 2\sqrt{t}$.
5) $|\tilde{u}_{q+1}| \le 2|u_t|$.

Suppose 4) holds. Then we deduce from (A21) and $s \ge q+1 \ge Q+1$ that the quantity on the left side of inequality (A18) is less than or equal to $2\sqrt{2}/[\sqrt{Q+1}-1]$. We can then conclude that

$$\omega_{\tilde{\pi}}(Q) \le \frac{3}{\sqrt{Q+1}-1} \quad \text{if 4) holds.} \tag{A24}$$

Now suppose 5) holds. Then the quantity on the left in (A18) is less than or equal to

$$2\left[\frac{|u_t|}{|u_1| + \cdots + |u_{t-1}|}\right]\left[\frac{|u_1| + \cdots + |u_{t-1}|}{|u_1| + \cdots + |u_{s-1}|}\right]. \tag{A25}$$

Since $t \ge q+1$, the quantity in brackets on the left in (A25) is less than or equal to $\omega_{\pi_\sigma}(Q)$. The quantity in brackets on the right in (A25) is less than or equal to the left-most term in the following string of inequalities:

$$\frac{|u_1| + \cdots + |u_{s-1}| + \sqrt{t}}{|u_1| + \cdots + |u_{s-1}|} \le 1 + \frac{\sqrt{t}}{s-1} \le 1 + \frac{\sqrt{2}}{\sqrt{s}-1} \le \frac{9}{2}.$$

Therefore

$$\omega_{\tilde{\pi}}(Q) \le 9\omega_{\pi_\sigma}(Q) \quad \text{if 5) holds.} \tag{A26}$$

Combining (A24) and (A26), we obtain (A18).

We can now say that $\sigma' \in \Sigma$. By construction, σ' is subordinate to σ and Property 3.1 is valid. Hence, $\sigma' \in \Sigma'$, and the proof is complete.

ACKNOWLEDGMENT

The first author wishes to thank J. Ziv for reminding him of the individual sequence x^* in the discussion of Question 1 in Section I.

REFERENCES

[1] A. Barron, "Logically smooth density estimation," Ph.D. dissertation, Stanford University, Stanford, CA, 1985.
[2] Ya. M. Barzdin, "Complexity of programs to determine whether natural numbers not greater than n belong to a recursively enumerable set," *Sov. Math.– Dokl.*, vol. 9, pp. 1251–1254, 1968.
[3] G. Boolos and R. Jeffrey, *Computability and Logic* 3rd ed. Cambridge, England: Cambridge Univ. Press, 1989.
[4] G. Chaitin, "On the length of programs for computing binary sequences," *J. Assoc. Comput. Mach.*, vol. 13, pp. 547–569, 1966.
[5] J. Kieffer, "Sample converses in source coding theory," *IEEE Trans. Inform. Theory*, vol. 37, pp. 263–268, 1991.
[6] A. Kolmogorov, "Three approaches to the quantitative definition of information," *Probl. Inform. Transm.*, vol. 1, pp. 4–7, 1965.
[7] A. Lempel and J. Ziv, "On the complexity of finite sequences," *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 75–81, 1976.
[8] S. Leung-Yan-Cheong and T. Cover, "Some equivalences between Shannon entropy and Kolmogorov complexity," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 331–338, 1978.
[9] M. Li and P. Vitanyi, *An Introduction to Kolmogorov Complexity and Its Applications*. New York: Springer-Verlag, 1993.
[10] R. Solomonoff, "A formal theory of inductive inference," *Inform. Contr.*, vol. 7, pp. 1–22 and 224–254, 1964.
[11] E. Yang and S. Shen, "Chaitin complexity, Shannon information content of a single event and infinite random sequences (I)," *Sci. in China*, ser. A, vol. 34, pp. 1183–1193, 1991.
[12] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Trans. Inform. Theory*, vol. IT-23, pp. 337–343, 1977.
[13] ——, "Compression of individual sequences via variable rate coding," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 530–536, 1978.
[14] A. Zvonkin and L. Levin, "The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms," *Russ. Math. Surveys*, vol. 25, pp. 83–124, 1970.