Correspondence

Meaningful Information

Paul M. Vitányi

***Abstract*—The information in an individual finite object (like a binary string) is commonly measured by its Kolmogorov complexity. One can divide that information into two parts: the information accounting for the useful regularity present in the object and the information accounting for the remaining accidental information. There can be several ways (model classes) in which the regularity is expressed. Kolmogorov has proposed the model class of finite sets, generalized later to computable probability mass functions. The resulting theory, known as Algorithmic Statistics, analyzes the algorithmic sufficient statistic when the statistic is restricted to the given model class. However, the most general way to proceed is perhaps to express the useful information as a total recursive function. The resulting measure has been called the "sophistication" of the object. We develop the theory of recursive functions statistic, the maximum and minimum value, the existence of absolutely nonstochastic objects (that have maximal sophistication—all the information in them is meaningful and there is no residual randomness), determine its relation with the more restricted model classes of finite sets, and computable probability distributions, in particular with respect to the algorithmic (Kolmogorov) minimal sufficient statistic, the relation to the halting problem and further algorithmic properties.**

***Index Terms*—Computability, constrained best fit model selection, Kolmogorov complexity, Kolmogorov structure function, lossy compression, minimal sufficient statistic, nonprobabilistic statistics, sophistication, sufficient statistic.**

I. Introduction

The information contained by an individual finite object (like a finite binary string) is objectively measured by its Kolmogorov complexity—the length of the shortest binary program that computes the object. Such a shortest program contains no redundancy: every bit is information; but is it meaningful information? If we flip a fair coin to obtain a finite binary string, then with overwhelming probability that string constitutes its own shortest program. However, also with overwhelming probability, all the bits in the string are meaningless information, random noise. On the other hand, let an object x be a sequence of observations of heavenly bodies. Then x can be described by the binary string pd, where p is the description of the laws of gravity, and the observational parameter setting, while d is the data-to-model code accounting for the (presumably Gaussian) measurement error in the data. This way we can divide the information in x into meaningful information p and data-to-model information d.

The main task for statistical inference and learning theory is to distil the meaningful information present in the data. The question arises whether it is possible to separate meaningful information from accidental information, and if so, how.

Manuscript received July 26, 2002; revised June 13, 2006. This work was supported in part by the EU Fifth Framework project QAIP, IST-1999-11234, the NoE QUIPROCONE IST-1999-29064, the ESF QiT Programmme, the EU Fourth Framework BRA NeuroCOLT II Working Group EP 27150, and the EU NoE PASCAL. The material of this correspondence was presented in part at the 13th International Symposium on Algorithms and Computation, Vancouver, BC, Canada, November 2002.

The author is with the CWI, 1098 SJ Amsterdam, The Netherlands (e-mail: Paul.Vitanyi@cwi.nl).

Communicated by M. J. Weinberger, Associate Editor for Source Coding.

Digital Object Identifier 10.1109/TIT.2006.881729

In statistical theory, every function of the data is called a "statistic" of the data. A central notion in probabilistic statistics is that of a "sufficient" statistic, introduced by the father of statistics R. A. Fisher [4]: "The statistic chosen should summarise the whole of the relevant information supplied by the sample. This may be called the Criterion of Sufficiency … In the case of the normal curve of distribution it is evident that the second moment is a sufficient statistic for estimating the standard deviation." For traditional problems, dealing with frequencies over small sample spaces, this approach is appropriate. But for current novel applications, average relations are often irrelevant, since the part of the support of the probability density function that will ever be observed has about zero measure. This is the case in, for example, complex video and sound analysis. There arises the problem that for individual cases the selection performance may be bad although the performance is good on average. There is also the problem of what probability means, whether it is subjective, objective, or exists at all.

To simplify matters, and because all discrete data can be binary coded, we consider only data samples that are finite binary strings. The basic idea is to base statistical theory on finite combinatorial principles independent of probabilistic assumptions, as the relation between the individual data and its explanation (model). We study extraction of meaningful information in an initially limited setting where this information be represented by a finite set (a model) of which the object (the data sample) is a typical member. Using the theory of Kolmogorov complexity, we can rigorously express and quantify typicality of individual objects. But typicality in itself is not necessarily a significant property: every object is typical in the singleton set containing only that object. More important is the following Kolmogorov complexity analog of probabilistic minimal sufficient statistic which implies typicality: The two-part description consisting of the description of the largest finite set, together with the index of the object in that set, is as concise as the shortest one-part description of the object. The finite set models the regularity present in the object (since it is a typical element of the set). This approach has been generalized to computable probability mass functions. The combined theory has been developed in detail in [6] and called "Algorithmic Statistics."

Here we study the most general form of algorithmic statistic: recursive function models. In this setting, the issue of meaningful information versus accidental information is put in its starkest form; and in fact, has been around for a long time in various imprecise forms unconnected with the sufficient statistic approach: The issue has sparked the imagination and entered scientific popularization in [8] as "effective complexity" (here "effective" is apparently used in the sense of "producing an effect" rather than "constructive" as is customary in the theory of computation). It is time that it receives formal treatment. Formally, we study the minimal length of a total recursive function that leads to an optimal length two-part code of the object being described. ("total" means the function value is defined for all arguments in the domain, and "partial" means that the function is possibly not total.) This minimal length has been called the "sophistication" of the object in [14], [15] in a different, but related, setting of compression and prediction properties of infinite sequences. That treatment is technically sufficiently vague so as to have no issue for the present work. We develop the notion based on prefix Turing machines, rather than on a variety of monotonic Turing machines as in the cited papers. Below we describe related work in detail and summarize our results. Subsequently, we formulate our problem in the formal setting of computable two-part codes.

0018-9448/$20.00 © 2006 IEEE

A. Related Work

A.N. Kolmogorov in 1974 [11] proposed an approach to a non-probabilistic statistics based on Kolmogorov complexity. An essential feature of this approach is to separate the data into meaningful information (a model) and meaningless information (noise). Cover [2], [3] attached the name "sufficient statistic" to a model of which the data is a "typical" member. In Kolmogorov's initial setting the models are finite sets. As Kolmogorov himself pointed out, this is no real restriction: the finite sets model class is equivalent, up to a logarithmic additive term, to the model class of computable probability density functions, as studied in [19], [20], [23]. Related aspects of "randomness deficiency" were formulated in [12], [13] and studied in [19], [24]. Despite its evident epistemological prominence in the theory of hypothesis selection and prediction, only selected aspects of the theory were studied in these references. Recent work [6] can be considered as a comprehensive investigation into the sufficient statistic for finite set models and computable probability density function models. Here we extend the approach to the most general form: the model class of total recursive functions. This idea was pioneered by [14], [15] who, unaware of a statistic connection, coined the cute word "sophistication." The algorithmic (minimal) sufficient statistic was related to an applied form in [7], [23]: the well-known "minimum description length" principle [1] in statistics and inductive reasoning.

In another paper [21] (chronologically following the present paper) we comprehensively treated all stochastic properties of the data in terms of Kolmogorov's so-called structure functions. The sufficient statistic aspect, studied here, covers only part of these properties. The results on the structure functions, including (non)computability properties, are valid, up to logarithmic additive terms, also for the model class of total recursive functions, as studied here.

B. This Work

It will be helpful for the reader to be familiar with initial parts of [6]. In [11], Kolmogorov observed that randomness of an object in the sense of having high Kolmogorov complexity is being random in just a "negative" sense. That being said, we define the notion of sophistication (minimal sufficient statistic in the total recursive function model class). It is demonstrated to be meaningful (existence and nontriviality). We then establish lower and upper bounds on the sophistication, and we show that there are objects for which the sophistication achieves the upper bound. In fact, these are objects in which all information is meaningful and there is (almost) no accidental information. That is, the simplest explanation of such an object is the object itself. In the simpler setting of finite set statistic the analogous objects were called "absolutely nonstochastic" by Kolmogorov. If such objects have high Kolmogorov complexity, then they can only be a random outcome of a "complex" random process, and Kolmogorov questioned whether such random objects, being random in just this "negative" sense, can occur in nature. But there are also objects that are random in the sense of having high Kolmogorov complexity, but simultaneously are typical outcomes of "simple" random processes. These were therefore said to be random in a "positive" sense [11]. An example are the strings of maximal Kolmogorov complexity; those are very unsophisticated (with sophistication about 0), and are typical outcomes of tosses with a fair coin—a very simple random process. We subsequently establish the equivalence between sophistication and the algorithmic minimal sufficient statistics of the finite set class and the probability mass function class. Finally, we investigate the algorithmic properties of sophistication: nonrecursiveness, upper semicomputability, and intercomputability relations of Kolmogorov complexity, sophistication, halting sequence.

II. Preliminaries

A *string* is a finite binary sequence, an element of $\{0,1\}^*$. If x is a string then the *length* $l(x)$ denotes the number of bits in x. We identify $\mathcal{N}$, the natural numbers, and $\{0,1\}^*$ according to the correspondence

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

Here ϵ denotes the *empty word*. Thus, $l(\epsilon)=0$. The emphasis is on binary sequences only for convenience; observations in any alphabet can be encoded in such a way that is "theory neutral." Below we will use the natural numbers and the strings interchangeably.

A string y is a *proper prefix* of a string x if we can write $x=yz$ for $z\neq\epsilon$. 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 *codewords*. An example of a prefix code, that is useful later, encodes the source word $x=x_1x_2\cdots x_n$ by the codeword

$$\overline{x}=1^n0x.$$

This prefix-free code is called *self-delimiting*, because there is fixed computer program associated with this code that can determine where the codeword $\bar{x}$ ends by reading it from left to right without backing up. This way, a composite code message can be parsed in its constituent codewords in one pass, by the computer program. (This desirable property holds for every prefix-free encoding of a finite set of source words, but not for every prefix-free encoding of an infinite set of source words. For a single finite computer program to be able to parse a code message, the encoding needs to have a certain uniformity property like the $\overline{x}$ code.) Since we use the natural numbers and the strings interchangeably, $l(\bar{x})$ where x is ostensibly an integer, means the length in bits of the self-delimiting code of the string with index x. On the other hand, $\overline{l(x)}$ where x is ostensibly a string, means the self-delimiting code of the string with index the length $l(x)$ of x. Using this code, we define the standard self-delimiting code for x to be $x'=\overline{l(x)}x$. It is easy to check that $l(\overline{x})=2n+1$ and $l(x')=n+2\log n+1$. Let $\langle\cdot\rangle$ denote a standard invertible effective one–one encoding from $\mathcal{N}\times\mathcal{N}$ to a subset of $\mathcal{N}$. For example, we can set $\langle x,y\rangle=x'y$ or $\langle x,y\rangle=\bar{x}y$. We can iterate this process to define $\langle x,\langle y,z\rangle\rangle$, and so on.

A. Kolmogorov Complexity

For definitions, notation, and an introduction to Kolmogorov complexity, see [16]. Informally, the Kolmogorov complexity, or algorithmic entropy, $K(x)$ of a string x is the length (number of bits) of a shortest binary program (string) to compute x on a fixed reference universal computer (such as a particular universal Turing machine). Intuitively, $K(x)$ represents the minimal amount of information required to generate x by any effective process. The conditional Kolmogorov complexity $K(x|y)$ of x relative to y is defined similarly as the length of a shortest program to compute x, if y is furnished as an auxiliary input to the computation. For technical reasons, we use a variant of complexity, the so-called prefix complexity, which is associated with Turing machines for which the set of programs resulting in a halting computation is prefix free. We realize prefix complexity by considering a special type of Turing machine with a one-way input tape, a separate work tape, 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*.

Definition 2.1: A function f from the natural numbers to the natural numbers is *partial recursive*, or *computable*, if there is a Turing machine T that computes it: $f(x)=T(x)$ for all x for which either f

or T (and hence both) are defined. This definition can be extended to (multi-tuples of) rational arguments and values.

Let $T_1, T_2, \ldots$ be a standard enumeration of all prefix Turing machines with a binary input tape, for example, the lexicographical length-increasing ordered syntactic prefix Turing machine descriptions [16], and let $\phi_1, \phi_2, \ldots$ be the enumeration of corresponding functions that are computed by the respective Turing machines (T_i computes ϕ_i). These functions are the partial recursive functions of effectively prefix-free encoded arguments. The Kolmogorov complexity of x is the length of the shortest binary program from which x is computed by such a function.

Definition 2.2: The prefix Kolmogorov complexity of x is

$$K(x) = \min_{p,i}\{l(\bar{i}) + l(p) : T_i(p) = x\}, \tag{II.1}$$

where the minimum is taken over $p \in \{0,1\}^*$ and $i \in \{1, 2, \ldots\}$. For the development of the theory we actually require the Turing machines to use *auxiliary* (also called *conditional*) information, by equipping the machine with a special read-only auxiliary tape containing this information at the outset. Then, the *conditional version* $K(x \mid y)$ of the prefix Kolmogorov complexity of x given y (as auxiliary information) is defined similarly as before, and the unconditional version is set to $K(x) = K(x \mid \epsilon)$.

Notation 2.3: From now on, we will denote by $\stackrel{+}{<}$ an inequality to within an additive constant, and by $\stackrel{\pm}{=}$ the situation when both $\stackrel{+}{<}$ and $\stackrel{+}{>}$ hold.

B. Two-Part Codes

Let $T_1, T_2, \ldots$ be the standard enumeration of Turing machines, and let U be a standard universal Turing machine satisfying $U(\langle i, p\rangle) = T_i(p)$ for all indices i and programs p. We fix U once and for all and call it the *reference universal prefix Turing machine*. The shortest program to compute x by U is denoted as x^* (if there is more than one of them, then x^* is the first one in standard enumeration). It is a deep and useful fact that the shortest effective description of an object x can be expressed in terms of a *two-part code*: the first part describing an appropriate Turing machine and the second part describing the program that interpreted by the Turing machine reconstructs x. The essence of the theory is the Invariance Theorem that can be informally stated as follows: For convenience, in the sequel we simplify notation and write $U(x, y)$ for $U(\langle x, y\rangle)$. Rewrite

$$\begin{aligned} K(x) &= \min_{p,i}\{l(\bar{i}) + l(p) : T_i(p) = x\} \\ &= \min_{p,i}\{2l(i) + l(p) + 1 : T_i(p) = x\} \\ &\le \min_{q}\{l(q) : U(\epsilon, q) = x\} + 2l(u) + 1 \\ &\le \min_{r,j}\{l(j^*) + l(r) : U(\epsilon, j^*\alpha r) = T_j(r) = x\} + 2l(u) + 1 \\ &\stackrel{+}{<} K(x). \end{aligned}$$

Here, the minima are taken over $p, q, r \in \{0,1\}^*$ and $i, j \in \{1, 2, \ldots\}$. The last equalities are obtained by using the universality of $U = T_u$ with $l(u) \stackrel{+}{=} 0$, and the explanation of the role of α follows in the next paragraph. As consequence

$$K(x) \stackrel{+}{=} \min_{r,j}\{l(j^*) + l(r) : U(\epsilon, j^*\alpha r) = T_j(r) = x\}$$

$$K(x) \stackrel{+}{=} K_U(x) = \min\{l(q) : U(\epsilon, q) = x\}.$$

Thus, $K(x)$ and $K_U(x)$ differ by at most an additive constant depending on the choice of U. It is standard to use

$$K(x) \equiv K_U(x) \tag{II.2}$$

instead of (II.1) as the definition of prefix Kolmogorov complexity [16]. However, we highlighted definition (II.1) to bring out the two-part code nature. By universal logical principles, the resulting theory is recursively invariant under adopting either definition (II.1) or definition (II.2), as long as we stick to one choice. If T stands for a literal description of the prefix Turing machine T in standard format, for example the index j when $T = T_j$, then we can write $K(T) \stackrel{+}{=} K(j)$. The string j^* is a shortest self-delimiting program of $K(j)$ bits from which U can compute j, and subsequent execution of the next self-delimiting fixed program α will compute $\bar{j}$ from j, and compute $U(j, r)$. Altogether, this has the effect that $U(\epsilon, j^*\alpha r) = T_j(r)$. If (j_0, r_0) minimizes the expression above, then $T_{j_0}(r_0) = x$, and hence $K(x|j_0) \stackrel{+}{<} l(r_0)$, and $K(j_0) + l(r_0) \stackrel{+}{=} K(x)$. It is straightforward that

$$K(j_0) + K(x|j_0) \stackrel{+}{>} K(x, j_0) \stackrel{+}{>} K(x)$$

and, therefore, we have $l(r_0) \stackrel{+}{<} K(x|j_0)$. Altogether, $l(r_0) \stackrel{\pm}{=} K(x|j_0)$. Replacing the minimizing $j = j_0$ by the minimizing $T = T_{j_0}$ and $l(r_0)$ by $K(x|T)$, we can rewrite the last displayed equation as

$$K(x) \stackrel{+}{=} \min_{T}\{K(T) + K(x \mid T) : T \in \{T_0, T_1, \ldots\}\}. \tag{II.3}$$

C. Meaningful Information

Expression (II.3) emphasizes the two-part code nature of Kolmogorov complexity: using the regular aspects of x to maximally compress. Suppose we consider an ongoing time-series $0101\ldots$ and we randomly stop gathering data after having obtained the initial segment

$$x = 10101010101010101010101010.$$

We can encode this x by a small Turing machine representing "the repeating pattern is 01," and which computes x, for example, from the program "13." Intuitively, the Turing machine part of the code squeezes out the *regularities* in x. What is left are irregularities, or *random aspects* of x relative to that Turing machine. The minimal-length two-part code squeezes out regularity only insofar as the reduction in the length of the description of random aspects is greater than the increase in the regularity description. In this setup the number of repetitions of the significant pattern is viewed as the random part of the data.

This interpretation of $K(x)$ as the shortest length of a two-part code for x, one part describing a Turing machine, or *model*, for the *regular* aspects of x and the second part describing the *irregular* aspects of x in the form of a program to be interpreted by T, has profound applications.

The "right model" is a Turing machine T among the ones that halt for all inputs, a restriction that is justified later, and reach the minimum description length in (II.3). This T embodies the amount of useful information contained in x. It remains to decide which such T to select among the ones that satisfy the requirement. Following Occam's Razor we opt here for the shortest one—a formal justification for this choice is given in [23]. The main problem with our approach is how to properly define a shortest program x^* for x that divides into parts $x^* = pq$ such that p represents an appropriate T.

D. Symmetry of Information

The following central notions are used in this correspondence. The *information in* x about y is $I(x : y) = K(y) - K(y \mid x^*)$. By the symmetry of information, a deep result of [5]

$$K(x, y) \stackrel{+}{=} K(x) + K(y \mid x^*) \stackrel{+}{=} K(y) + K(x \mid y^*). \tag{II.4}$$

Rewriting according to symmetry of information we see that $I(x:y) \stackrel{+}{=} I(y:x)$ and, therefore, we call the quantity $I(x:y)$ the *mutual information* between x and y.

III. Model Classes

Instead of the model class of finite sets, or computable probability density functions, as in [6], in this work we focus on the most general form of algorithmic model class: total recursive functions. We define the different model classes and summarize the central notions of "randomness deficiency" and "typicality" for the canonical finite set models to obtain points of reference for the related notions in the more general model classes.

A. Set Models

The model class of *finite sets* consists of the set of finite subsets $S \subseteq \{0,1\}^*$. The complexity of the finite set S is $K(S)$—the length (number of bits) of the shortest binary program p from which the reference universal prefix machine U computes a listing of the elements of S and then halts. That is, if $S = \{x_1, \ldots, x_n\}$, then $U(p) = \langle x_1, \langle x_2, \ldots, \langle x_{n-1}, x_n \rangle \ldots \rangle \rangle$. The *conditional complexity* $K(x \mid S)$ of x, given S, is the length (number of bits) in the shortest binary program p from which the reference universal prefix machine U, given S literally as auxiliary information, computes x. For every finite set $S \subseteq \{0,1\}^*$ containing x we have

$$K(x \mid S) \stackrel{+}{<} \log |S|. \tag{III.1}$$

Indeed, consider the self-delimiting code of x consisting of its $\lceil \log |S| \rceil$ bit long index of x in the lexicographical ordering of S. This code is called *data-to-model code*. Its length quantifies the maximal "typicality," or "randomness," data (possibly different from x) can have with respect to this model. The lack of typicality of x with respect to S is measured by the amount by which $K(x \mid S)$ falls short of the length of the data-to-model code, the *randomness deficiency* of x in S, defined by

$$\delta(x \mid S) = \log |S| - K(x \mid S) \tag{III.2}$$

for $x \in S$, and ∞ otherwise. Data x is typical with respect to a finite set S if the randomness deficiency is small. If the randomness deficiency is close to 0, then there are no simple special properties that single it out from the majority of elements in S. This is not just terminology. Let $S \subseteq \{0,1\}^n$. According to common viewpoints in probability theory, each property represented by S defines a large subset of S consisting of elements having that property, and, conversely, each large subset of S represents a property. For probabilistic ensembles we take high probability subsets as properties; the present case is uniform probability with finite support. For some appropriate fixed constant c, let us identify a property represented by S with a subset S' of S of cardinality $|S'| > (1 - 1/c)|S|$. If $\delta(x \mid S)$ is close to 0, then x satisfies (that is, is an element of) *all* properties (that is, sets) $S' \subseteq S$ of low Kolmogorov complexity $K(S') = O(\log n)$. The precise statements and quantifications are given in [16], [21], and we do not repeat them here.

B. Probability Models

The model class of computable probability density functions consists of the set of functions $P : \{0,1\}^* \to [0,1]$ with $\sum P(x) = 1$. "Computable" means here that there is a Turing machine T_P that, given x and a positive rational ϵ, computes $P(x)$ with precision ϵ. The (prefix-) complexity $K(P)$ of a computable (possibly partial) function P is defined by

$$K(P) = \min_i \{K(i) : \text{Turing machine } T_i \text{ computes } P\}.$$

C. Function Models

The model class of total recursive functions consists of the set of functions $f : \{0,1\}^* \to \{0,1\}^*$ such that there is a Turing machine T such that $T(i) < \infty$ and $f(i) = T(i)$, for every $i \in \{0,1\}^*$. The (prefix-) complexity $K(f)$ of a total recursive function f is defined by

$$K(f) = \min_i \{K(i) : \text{Turing machine } T_i \text{ computes } f\}.$$

If f^* is a shortest program for computing the function f (if there is more than one of them then f^* is the first one in enumeration order), then $K(f) = l(f^*)$.

Remark 3.1: In the definitions of $K(P)$ and $K(f)$, the objects being described are functions rather than finite binary strings. To unify the approaches, we can consider a finite binary string x as corresponding to a function having value x for argument 0. Note that we can upper-semi-compute x^* given x, but we cannot upper-semi-compute P^* given P (as an oracle), or f^* given f (again given as an oracle), since we should be able to verify agreement of a program for a function and an oracle for the target function, on all infinitely many arguments. ♢

IV. Typicality

To explain typicality for general model classes, it is convenient to use the distortion-rate [17], [18] approach for individual data recently introduced in [9], [22]. Modeling the data can be viewed as encoding the data by a model: the data are source words to be coded, and models are codewords for the data. As before, the set of possible data is $\mathcal{D} = \{0,1\}^*$. Let $\mathcal{R}^+$ denote the set of nonnegative real numbers. For every model class $\mathcal{M}$ (particular set of codewords) we choose an appropriate recursive function $d : \mathcal{D} \times \mathcal{M} \to \mathcal{R}^+$ defining the *distortion* $d(x, M)$ between data x and model M.

Remark 4.1: The choice of distortion function is a selection of which aspects of the data are relevant, or meaningful, and which aspects are irrelevant (noise). We can think of the distortion as measuring how far the model falls short in representing the data. Distortion-rate theory underpins the practice of lossy compression. For example, lossy compression of a sound file gives as "model" the compressed file where, among others, the very high and very low inaudible frequencies have been suppressed. Thus, the distortion function will penalize the deletion of the inaudible frequencies but lightly because they are not relevant for the auditory experience. ♢

Example 4.2: Let us look at various model classes and distortion measures:

i) The set of models are the finite sets of finite binary strings. Let $S \subseteq \{0,1\}^*$ and $|S| < \infty$. We define $d(x, S) = \log |S|$ if $x \in S$, and ∞ otherwise.

ii) The set of models are the computable probability density functions P mapping $\{0,1\}^*$ to $[0,1]$. We define $d(x, S) = \log 1/P(x)$ if $P(x) > 0$, and ∞ otherwise.

iii) The set of models are the total recursive functions f mapping $\{0,1\}^*$ to $\mathcal{N}$. We define $d(x, f) = \min\{l(d) : f(d) = x\}$, and ∞ if no such d exists. ♢

If $\mathcal{M}$ is a model class, then we consider *distortion balls* of given radius r centered on $M \in \mathcal{M}$

$$B_M(r) = \{y : d(y, M) \leq r\}.$$

This way, every model class and distortion measure can be treated similarly to the canonical finite set case, which, however, is especially simple in that the radius is not variable. That is, there is only one distortion ball centered on a given finite set, namely, the one with radius

equal to the log-cardinality of that finite set. In fact, that distortion ball equals the finite set on which it is centered.

Let $\mathcal{M}$ be a model class and d a distortion measure. Since in our definition the distortion is recursive, given a model $M \in \mathcal{M}$ and diameter r, the elements in the distortion ball of diameter r can be recursively enumerated from the distortion function. Giving the index of any element x in that enumeration we can find the element. Hence, $K(x|M,r) \stackrel{+}{<} \log|B_M(r)|$. On the other hand, the vast majority of elements y in the distortion ball have complexity $K(y|M,r) \stackrel{+}{>} \log|B_M(r)|$ since, for every constant c, there are only $2^{\log|B_M(r)|-c} - 1$ binary programs of length $< \log|B_M(r)| - c$ available, and there are $|B_M(r)|$ elements to be described. We can now reason as in the similar case of finite set models. With data x and $r = d(x,M)$, if $K(x|M, d(x,M)) \stackrel{+}{>} |B_M(d(x,M))|$, then x belongs to every large majority of element (has the property represented by that majority) of the distortion ball $B_M(d(x,M))$, provided that property is simple in the sense of having a description of low Kolmogorov complexity.

Definition 4.3: the *randomness deficiency* of x with respect to model M under distortion d is defined as

$$\delta(x \mid M) = \log|B_M(d(x,M))| - K(x|M, d(x,M)).$$

Data x is *typical* for model $M \in \mathcal{M}$ (and that model "typical" or "best fitting" for x) if

$$\delta(x \mid M) \stackrel{+}{=} 0. \tag{IV.1}$$

If x is typical for a model M, then the shortest way to effectively describe x, given M, takes about as many bits as the descriptions of the great majority of elements in a recursive enumeration of the distortion ball. So there are no special simple properties that distinguish x from the great majority of elements in the distortion ball: they are all typical or random elements in the distortion ball (that is, with respect to the contemplated model).

Example 4.4: Continuing Example 4.2 by applying (IV.1) to different model classes we get the following.

i) *Finite sets:* For finite set models S, clearly $K(x|S) \stackrel{+}{<} \log|S|$. Together with (IV.1) we have that x is typical for S, and S best fits x, if the randomness deficiency according to (III.2)satisfies $\delta(x|S) \stackrel{+}{=} 0$.

ii) *Computable probability density functions:* Instead of the data-to-model code length $\log|S|$ for finite set models, we consider the data-to-model code length $\log 1/P(x)$ (the Shannon–Fano code). The value $\log 1/P(x)$ measures how likely x is under the hypothesis P. For probability models P, define the conditional complexity $K(x \mid P, \lceil \log 1/P(x) \rceil)$ as follows. Say that a function A approximates P if $|A(y,\epsilon) - P(y)| < \epsilon$ for every y and every positive rational ϵ. Then $K(x \mid P, \lceil \log 1/P(x) \rceil)$ is defined as the minimum length of a program that, given $\lceil \log 1/P(x) \rceil$ and any function A approximating P as an oracle, prints x.

Clearly, $K(x|P, \lceil \log 1/P(x) \rceil) \stackrel{+}{<} \log 1/P(x)$. Together with (IV.1), we have that x is typical for P, and P best fits x, if

$$K(x|P, \lceil \log 1/P(x) \rceil) \stackrel{+}{>} \log|\{y : \log 1/P(y) \le \log 1/P(x)\}|.$$

The right-hand side set condition is the same as $P(y) \ge P(x)$, and there can be only $\le 1/P(x)$ such y, since otherwise the total probability exceeds 1. Therefore, the requirement, and hence typicality, is implied by $K(x|P, \lceil \log 1/P(x) \rceil) \stackrel{+}{>} \log 1/P(x)$. Define the randomness deficiency by

$$\delta(x \mid P) = \log 1/P(x) - K(x \mid P, \lceil \log 1/P(x) \rceil).$$

Altogether, a string x is typical for a distribution P, or P is the best fitting model for x, if $\delta(x \mid P) \stackrel{+}{=} 0$.

iii) *Total recursive functions:* In place of $\log|S|$ for finite set models we consider the data-to-model code length (actually, the distortion $d(x,f)$ above)

$$l_x(f) = \min\{l(d) : f(d) = x\}.$$

Define the conditional complexity $K(x \mid f, l_x(f))$ as the minimum length of a program that, given $l_x(f)$ and an oracle for f, prints x.

Clearly, $K(x|f, l_x(f)) \stackrel{+}{<} l_x(f)$. Together with (IV.1), we have that x is typical for f, and f best fits x, if

$$K(x|f, l_x(f)) \stackrel{+}{>} \log\{y : l_y(f) \le l_x(f)\}.$$

There are at most $(2^{l_x(f)+1} - 1)$—many y satisfying the set condition since $l_y(f) \in \{0,1\}^*$. Therefore, the requirement, and hence typicality, is implied by $K(x|f, l_x(f)) \stackrel{+}{>} l_x(f)$. Define the randomness deficiency by $\delta(x \mid f) = l_x(f) - K(x \mid f, l_x(f))$. Altogether, a string x is typical for a total recursive function f, and f is the best fitting recursive function model for x if $\delta(x \mid f) \stackrel{+}{=} 0$, or written differently

$$K(x|f, l_x(f)) \stackrel{+}{=} l_x(f). \tag{IV.2}$$

Note that since $l_x(f)$ is given as conditional information, with $l_x(f) = l(d)$ and $f(d) = x$, the quantity $K(x|f, l_x(f))$ represents the number of bits in a shortest *self-delimiting* description of d. ♢

Remark 4.5: We required $l_x(f)$ in the conditional in (IV.2). This is the information about the radius of the distortion ball centered on the model concerned. Note that in the canonical finite set model case, as treated in [11], [6], [21], every model has a fixed radius which is explicitly provided by the model itself. But in the more general model classes of computable probability density functions, or total recursive functions, models can have a variable radius. There are subclasses of the more general models that have fixed radiuses (like the finite set models).

i) In the computable probability density functions, one can think of the probabilities with a finite support, for example, $P_n(x) = 1/2^n$ for $l(x) = n$, and $P(x) = 0$ otherwise.

ii) In the total recursive function case, one can similarly think of functions with finite support, for example, $f_n(x) = \sum_{i=1}^n x_i$ for $x = x_1 \ldots x_n$, and $f_n(x) = 0$ for $l(x) \neq n$.

The incorporation of te radius in the model will increase the complexity of the model, and hence of the minimal sufficient statistic below. ♢

V. Sufficient Statistic

A *statistic* is a function mapping the data to an element (model) in the contemplated model class. With some sloppiness of terminology, we often call the function value (the model) also a statistic of the data. The most important concept in this correspondence is the sufficient statistic. For an extensive discussion of this notion for specific model classes see [6], [21]. A statistic is called sufficient if the two-part description of the data by way of the model and the data-to-model code is as concise as the shortest one-part description of x. Consider a model class $\mathcal{M}$.

Definition 5.1: A model $M \in \mathcal{M}$ is a *sufficient statistic* for x if

$$K(M, d(x,M)) + \log|B_M(d(x,M))| \stackrel{+}{=} K(x). \tag{V.1}$$

Lemma 5.2: If M is a sufficient statistic for x, then

$$K(x \mid M, d(x,M) \stackrel{+}{=} \log|B_M(d(x,M))|$$

that is, x is typical for M.

Proof: We can rewrite

$$\begin{aligned} K(x) &\stackrel{+}{<} K(x, M, d(x, M)) \\ &\stackrel{+}{<} K(M, d(x, M)) + K(x|M, d(x, M)) \\ &\stackrel{+}{<} K(M, d(x, M)) + \log |B_M(d(x, M))| \stackrel{+}{=} K(x). \end{aligned}$$

The first three inequalities are straightforward and the last equality is by the assumption of sufficiency. Altogether, the first sum equals the second sum, which implies the lemma. □

Thus, if M is a sufficient statistic for x, then x is a typical element for M, and M is the best fitting model for x. Note that the converse implication, "typicality" implies "sufficiency," is not valid. Sufficiency is a special type of typicality, where the model does not add significant information to the data, since the preceding proof shows $K(x) \stackrel{+}{=} K(x, M, d(x, M))$. Using the symmetry of information (II.4) this shows that

$$K(M, d(x, M) \mid x) \stackrel{+}{=} K(M \mid x) \stackrel{+}{=} 0. \tag{V.2}$$

This means that we have the following.

i) A sufficient statistic M is determined by the data in the sense that we need only an $O(1)$-bit program, possibly depending on the data itself, to compute the model from the data.

ii) For each model class and distortion there is a universal constant c such that for every data item x there are at most c sufficient statistics.

Example 5.3: Finite sets: For the model class of finite sets, a set S is a sufficient statistic for data x if

$$K(S) + \log |S| \stackrel{+}{=} K(x).$$

Computable probability density functions: For the model class of computable probability density functions, a function P is a sufficient statistic for data x if

$$K(P) + \log 1/P(x) \stackrel{+}{=} 0. \qquad \diamondsuit$$

Definition 5.4: For the model class of total recursive functions, a function f is a *sufficient statistic* for data x if

$$K(x) \stackrel{+}{=} K(f) + l_x(f). \tag{V.3}$$

Following the above discussion, the meaningful information in x is represented by f (the model) in $K(f)$ bits, and the meaningless information in x is represented by d (the noise in the data) with $f(d) = x$ in $l(d) = l_x(f)$ bits. Note that $l(d) \stackrel{+}{=} K(d) \stackrel{+}{=} K(d|f^*)$, since the two-part code (f^*, d) for x cannot be shorter than the shortest one-part code of $K(x)$ bits, and therefore the d-part must already be maximally compressed. By Lemma 5.2, $l_x(f) \stackrel{+}{=} K(x \mid f^*, l_x(f))$, x is typical for f, and hence, $K(x) \stackrel{+}{=} K(f) + K(x \mid f^*, l_x(f))$.

VI. Minimal Sufficient Statistic

Definition 6.1: Consider the model class of total recursive functions. A *minimal sufficient statistic* for data x is a sufficient statistic (V.3) for x of minimal prefix complexity. Its length is known as the *sophistication* of x, and is defined by

$$\mathrm{soph}(x) = \min\{K(f) : K(f) + l_x(f) \stackrel{+}{=} K(x)\}.$$

Recall that the *reference* universal prefix Turing machine U was chosen such that $U(T, d) = T(d)$ for all T and d. Looking at it slightly more from a programming point of view, we can define a pair (T, d) to be a *description* of a finite string x, if $U(T, d)$ prints x and T is a Turing machine computing a function f so that $f(d) = x$. For the notion of minimal sufficient statistic to be nontrivial, it should be impossible to always shift, if $f(d) = x$ and $K(f) + l_x(f) \stackrel{+}{=} K(x)$ with $K(f) \stackrel{+}{\neq} 0$, information from f to d and write, for example, $f'(d') = x$ with $K(f') + l_x(f') \stackrel{+}{=} K(x)$ with $K(f') \stackrel{+}{=} 0$. If the model class contains a fixed universal model that can mimic all other models, then we can always shift all model information to the data-to-(universal)-model code. Note that this problem does not arise in common statistical model classes: these do not contain universal models in the algorithmic sense. First, we show that the partial recursive function model class, because it contains a universal element, does not allow a straightforward nontrivial division into meaningful and meaningless information.

Lemma 6.2: Assume for the moment that we allow all partial recursive programs as statistic. Then, the sophistication of all data x is $\stackrel{+}{=} 0$.

Proof: Let the index of U (the reference universal prefix Turing machine) in the standard enumeration $T_1, T_2, \ldots$ of prefix Turing machines be u. Let T_f be a Turing machine computing f. Suppose that $U(T_f, d) = x$. Then, also $U(u, \langle T_f, d\rangle) = U(T_f, d) = x$. □

Remark 6.3: This shows that unrestricted partial recursive statistics are uninteresting. Naively, this could leave the impression that the separation of the regular and the random part of the data is not as objective as the whole approach allows us to hope for. If we consider complexities of the minimal sufficient statistics in model classes of increasing power: finite sets, computable probability distributions, total recursive functions, partial recursive functions, then the complexities appear to become smaller all the time eventually reaching zero. It would seem that the universality of Kolmogorov complexity, based on the notion of partial recursive functions, would suggest a similar universal notion of sufficient statistic based on partial recursive functions. But in this case the very universality trivializes the resulting definition: because partial recursive functions contain a particular universal element that can simulate all the others, this implies that the universal partial recursive function is a universal model for all data, and the data-to-model code incorporates all information in the data. Thus, if a model class contains a universal model that can simulate all other models, then this model class is not suitable for defining two-part codes consisting of meaningful information and accidental information. It turns out that the key to nontrivial separation is the requirement that the program witnessing the sophistication be *total*. That the resulting separation is nontrivial is evidenced by the fact, shown below, that the amount of meaningful information in the data does not change by more than a logarithmic additive term under change of model classes among finite set models, computable probability models, and total recursive function models. That is, very different model classes all result in the same amount of meaningful information in the data, up to negligible differences. So if deterioration occurs in widening model classes, it occurs all at once by having a universal element in the model class. $\diamondsuit$

Apart from triviality, a class of statistics can also possibly be vacuous by having the length of the minimal sufficient statistic exceed $K(x)$. Our first task is to determine whether the definition is nonvacuous. We will distinguish sophistication in different description modes:

Lemma 6.4 (Existence): For every finite binary string x, the sophistication satisfies $\mathrm{soph}(x) \stackrel{+}{<} K(x)$.

Proof: By definition of the prefix complexity there is a program x^* of length $l(x^*) = K(x)$ such that $U(x^*, \epsilon) = x$. This program x^* can be partial. But we can define another program $x^*_s = sx^*$ where s is a program of a constant number of bits that tells the following program to ignore its actual input and compute as if its input were ϵ. Clearly, x^*_s

is total and is a sufficient statistic of the total recursive function type, that is,

$$\mathrm{soph}(x) \leq l(x_s^*) \stackrel{+}{<} l(x^*) = K(x). \qquad \square$$

The previous lemma gives an upper bound on the sophistication. This still leaves the possibility that the sophistication is always $\stackrel{+}{=} 0$, for example in the most liberal case of unrestricted totality. But this turns out to be impossible.

Theorem 6.5:

i) For every x, if a sufficient statistic f satisfies $K(l_x(f)|f^*) \stackrel{+}{=} 0$, then $K(f) \stackrel{+}{>} K(K(x))$ and $l_x(f) \stackrel{+}{<} K(x) - K(K(x))$.

ii) For x as a variable running through a sequence of finite binary strings of increasing length, we have

$$\liminf_{l(x)\to\infty} \mathrm{soph}(x) \stackrel{+}{=} 0. \tag{VI.1}$$

iii) For every n, there exists an x of length n, such that every sufficient statistic f for x that satisfies $K(l_x(f)|f^*) \stackrel{+}{=} 0$ has $K(f) \stackrel{+}{>} n$.

iv) For every n there exists an x of length n such that

$$\mathrm{soph}(x) \stackrel{+}{>} n - \log n - 2\log\log n.$$

Proof:

i) If f is a sufficient statistic for x, then

$$K(x) \stackrel{+}{=} K(f) + K(d \mid f^*) \stackrel{+}{=} K(f) + l_x(f). \tag{VI.2}$$

Since $K(l_x(f)|f^*) \stackrel{+}{=} 0$, given an $O(1)$ bit program q, we can retrieve both $l_x(f)$ and also $K(f) = l(f^*)$ from f^*. Therefore, we can retrieve $K(x) \stackrel{+}{=} K(f)+l_x(f)$ from qf^*. That shows that $K(K(x)) \stackrel{+}{<} K(f)$. This proves both the first statement, and the second statement follows by (VI.2).

ii) An example of very unsophisticated strings are the individually random strings with high complexity: x of length $l(x) = n$ with complexity $K(x) \stackrel{+}{=} n + K(n)$. Then, the *identity* program ι with $\iota(d) = d$ for all d is total, has complexity $K(\iota) \stackrel{+}{=} 0$, and satisfies $K(x) \stackrel{+}{=} K(\iota)+l(x^*)$. Hence, ι witnesses that $\mathrm{soph}(x) \stackrel{+}{=} 0$. This shows (VI.1).

iii) Consider the set $S^m = \{y : K(y) \leq m\}$. By [6] we have $\log|S^m| \stackrel{+}{=} m - K(m)$. Let $m \leq n$. Since there are 2^n strings of length n, there are strings of length n not in S^m. Let x be any such string, and denote $k = K(x)$. Then, by construction $k > m$ and by definition $k \stackrel{+}{<} n + K(n)$. Let f be a sufficient statistic for x. Then, $K(f) + l_x(f) \stackrel{+}{=} k$. By assumption, there is an $O(1)$-bit program q such that $U(qf^*) = l_x(f)$. Let d witness $l_x(f)$ by $f(d) = x$ with $l(d) = l_x(f)$. Define the set $D = \{0,1\}^{l_x(f)}$. Clearly, $d \in D$. Since x can be retrieved from f and the lexicographical index of d in D, and $\log|D| = l_x(f)$, we have $K(f) + \log|D| \stackrel{+}{=} k$. Since we can obtain D from qf^* we have $K(D) \stackrel{+}{<} K(f)$. On the other hand, since we can retrieve x from D and the index of d in D, we must have $K(D) + \log|D| \stackrel{+}{>} k$, which implies $K(D) \stackrel{+}{>} K(f)$. Altogether, therefore, $K(D) \stackrel{+}{=} K(f)$.

We now show that we can choose x so that $K(D) \stackrel{+}{>} n$, and therefore, $K(f) \stackrel{+}{>} n$. For every length n, there exists a z of complexity $K(z \mid n) \stackrel{+}{<} n$ such that a minimal sufficient finite set statistic S for z has complexity at least $K(S \mid n) \stackrel{+}{>} n$, by [6, Theorem IV.2]. Since $\{z\}$ is trivially a sufficient statistic for z, it follows that $K(z \mid n) \stackrel{+}{=} K(S \mid n) \stackrel{+}{=} n$. This implies $K(z), K(S) \stackrel{+}{>} n$. Therefore, we can choose $m = n - c_2$ for a large enough constant c_2 so as to ensure that $z \notin S^m$. Consequently, we can choose x above as such a z. Since every finite set sufficient statistic for x has complexity at least that of an finite set minimal sufficient statistic for x, it follows that $K(D) \stackrel{+}{>} n$. Therefore, $K(f) \stackrel{+}{>} n$, which was what we had to prove.

iv) In the proof of i) we used $K(l_x(f)|f^*) \stackrel{+}{=} 0$. Without using this assumption, the corresponding argument yields $k \stackrel{+}{<} K(f) + K(l_x(f))) + \log|D|$. We also have $K(f) + l_x(f) \stackrel{+}{<} k$ and $l(d) \stackrel{+}{=} \log|D|$. Since we can retrieve x from D and its index in D, the same argument as above shows $|K(f) - K(D)| \stackrel{+}{<} K(l_x(f))$, and still following the argument above, $K(f) \stackrel{+}{>} n - K(l_x(f))$. Since $l_x(f) \stackrel{+}{<} n$, we have $K(l_x(f)) \stackrel{+}{<} \log n + 2\log\log n$. This proves the statement. $\square$

The useful (V.2) states that there is a constant, such that for every x there are at most that constant many sufficient statistics for x, and there is a constant length program (possibly depending on x), that generates all of them from x^*. In fact, there is a slightly stronger statement from which this follows:

Lemma 6.6: There is a universal constant c, such that for every x, the number of f^*d such that $f(d) = x$ and $K(f) + l(d) \stackrel{+}{=} K(x)$, is bounded above by c.

Proof: Let the prefix Turing machine T_f compute f. Since $U(T_f, d) = x$ and $K(T_f) + l(d) \stackrel{+}{=} K(x)$, the combination f^*d (with self-delimiting f^*) is a shortest prefix program for x. From [16, Exercise 3.3.7 item (b) on p. 205], it follows that the number of shortest prefix programs is upper-bounded by a universal constant. $\square$

VII. Relation Between Sufficient Statistic for Different Model Classes

Earlier work studied sufficiency for finite set models, and computable probability mass functions models, [6]. The most general models that are still meaningful are total recursive functions as studied here. We show that there are corresponding, almost equivalent, sufficient statistics in all model classes.

Lemma 7.1:

i) If S is a sufficient statistic of x (finite set type), then there is a corresponding sufficient statistic P of x (probability mass function type) such that $K(P) \stackrel{+}{=} K(S), \log 1/P(x) \stackrel{+}{=} \log|S|$, and $K(P \mid x^*) \stackrel{+}{=} 0$.

ii) If P is a sufficient statistic of x of the computable total probability density function type, then there is a corresponding sufficient statistic f of x of the total recursive function type such that $K(f) \stackrel{+}{=} K(P)$, $l_x(f) \stackrel{+}{=} \log 1/P(x)$, and $K(f \mid x^*) \stackrel{+}{=} 0$.

Proof:

i) By assumption, S is a finite set such that $x \in S$ and $K(x) \stackrel{+}{=} K(S) + \log|S|$. Define the probability distribution $P(y) = 1/|S|$ for $y \in S$ and $P(y) = 0$ otherwise. Since S is finite, P is computable. Since $K(S) \stackrel{+}{=} K(P)$, and $\log|S| = \lceil \log 1/P(x) \rceil$, we have $K(x) \stackrel{+}{=} K(P) + \log 1/P(x)$. Since P is a computable probability mass function we have $K(x \mid P^*) \stackrel{+}{<} \log 1/P(x)$, by the standard Shannon–Fano code construction [3] that assigns a codeword of length $\log 1/P(x)$ to x. Since by (II.4) we have

$$K(x) \stackrel{+}{<} K(x,P) \stackrel{+}{=} K(P) + K(x \mid P^*)$$

it follows that $\log 1/P(x) \stackrel{+}{<} K(x \mid P^*)$. Hence,

$$K(x \mid P^*) \stackrel{+}{=} \log 1/P(x).$$

Therefore, by (II.4), $K(x,P) \stackrel{+}{=} K(x)$ and, by rewriting $K(x,P)$ in the other way according to (II.4), $K(P \mid x^*) \stackrel{+}{=} 0$.

ii) By assumption, P is a computable probability density function with $P(x) > 0$ and $K(x) \stackrel{+}{=} K(P) + \log 1/P(x)$. The witness of

this equality is a shortest program P^* for P and a codeword s_x for x according to the standard Shannon–Fano code, [3], with $l(s_x) \stackrel{+}{=} \log 1/P(x)$. Given P, we can reconstruct x from s_x by a fixed standard algorithm. Define the recursive function f from P such that $f(s_x) = x$. In fact, from P^* this only requires a constant length program q, so that $T_f = qP^*$ is a program that computes f in the sense that $U(T_f, d) = f(d)$ for all d. Similarly, P can be retrieved from f. Hence, $K(f) \stackrel{+}{=} K(P)$ and $K(x) \stackrel{+}{=} K(f) + l(s_x)$. That is, f is a sufficient statistic for x. Also, f is a total recursive function. Since $f(s_x) = x$ we have $K(x \mid f^*) \stackrel{+}{<} l(s_x)$, and $K(x \mid f^*) \stackrel{+}{<} l(s_x)$. This shows that $K(x) \stackrel{+}{>} K(f) + K(x \mid f^*)$, and since x can by definition be reconstructed from f^* and a program of length $K(x \mid f^*)$, it follows that equality must hold. Consequently, $l(s_x) \stackrel{+}{=} K(x \mid f^*)$, and hence, by (II.4), $K(x, f) \stackrel{+}{=} K(x)$ and $K(f \mid x^*) \stackrel{+}{=} 0$. □

We have now shown that a sufficient statistic in a less general model class corresponds directly to a sufficient statistic in the next more general model class. We now show that, with a negligible error term, a sufficient statistic in the most general model class of total recursive functions has a directly corresponding sufficient statistic in the least general finite set model class. That is, up to negligible error terms, a sufficient statistic in any of the model classes has a direct representative in any of the other model classes.

Lemma 7.2: Let x be a string of length n, and f be a total recursive function sufficient statistic for x. Then, there is a finite set $S \ni x$ such that $K(S) + \log |S| \stackrel{+}{=} K(x) + O(\log n)$.

Proof: By assumption, there is an $O(1)$-bit program q such that $U(qf^*) = l_x(f)$. For each $y \in \{0,1\}^*$, let $i_y = \min\{i : f(i) = y\}$. Define $S = \{y : f(i_y) = y, l(i_y) \stackrel{+}{<} l_x(f)\}$. We can compute S by computation of $f(i)$, on all arguments i of at most $l(i) \leq l_x(f)$ bits, since by assumption f is total. This shows

$$K(S) \stackrel{+}{<} K(f, l_x(f)) \stackrel{+}{<} K(f) + K(l_x(f)).$$

Since $l_x(f) \stackrel{+}{<} K(x)$, we have $l_x(f) \stackrel{+}{<} l(x) = n$. Moreover, $\log |S| \stackrel{+}{=} l_x(f)$. Since $x \in S$, $K(x) \stackrel{+}{<} K(S) + \log |S| \stackrel{+}{<} K(x) + O(\log n)$, where we use the sufficiency of f to obtain the last inequality. □

VIII. Algorithmic Properties

We investigate the recursion properties of the sophistication function. In [5], Gács gave an important and deep result, see (VIII.1) below, that quantifies the uncomputability of $K(x)$ (the bare uncomputability can be established in a much simpler fashion). For every length n there is an x of length n such that

$$\log n - \log\log n \stackrel{+}{<} K(K(x) \mid x) \stackrel{+}{<} \log n. \tag{VIII.1}$$

Note that the right-hand side holds for every x by the simple argument that $K(x) \leq n + 2\log n$ and hence, $K(K(x)) \stackrel{+}{<} \log n$. But there are x's such that the length of the shortest program to compute $K(x)$ almost reaches this upper bound, even if the full information about x is provided. It is natural to suppose that the sophistication function is not recursive either. The following lemmas suggest that the complexity function is more uncomputable than the sophistication.

Theorem 8.1: The function soph is not recursive.

Proof: Given n, let x_0 be the least x such that $\mathrm{soph}(x) > n - 2\log n$. By Theorem 6.5, we know that there exist x such that $\mathrm{soph}(x) \to \infty$ for $x \to \infty$, hence, x_0 exists. Assume by way of contradiction that the sophistication function is computable. Then, we can find x_0, given n, by simply computing the successive values of the function. But then $K(x_0) \stackrel{+}{<} K(n)$, while by Lemma 6.4, $K(x_0) \stackrel{+}{>} \mathrm{soph}(x_0)$ and by assumption $\mathrm{soph}(x_0) > n - 2\log n$, which is impossible. □

The *halting sequence* $\chi = \chi_1\chi_2\ldots$ is the infinite binary characteristic sequence of the halting problem, defined by $\chi_i = 1$ if the reference universal prefix Turing machine U halts on the ith input: $U(i) < \infty$, and 0 otherwise.

Lemma 8.2: Let f^* be a total recursive function sufficient statistic of x.

i) We can compute $K(x)$ from f^* and x, up to fixed constant precision, which implies that $K(K(x) \mid f^*, x) \stackrel{+}{=} 0$.

ii) If also $K(l_x(f)|f^*) \stackrel{+}{=} 0$, then we can compute $K(x)$ from f^*, up to fixed constant precision, which implies that $K(K(x) \mid f^*) \stackrel{+}{=} 0$.

Proof:

i) Since f is total, we can run $f(e)$ on all strings $e \leq x$ in lexicographical length-increasing order. Since f is total, we will find a shortest string e_0 such that $f(e_0) = x$. Set $l_x(f) = l(e_0)$. Since $l(f^*) = K(f)$, and by assumption, $K(f) + l_x(f) \stackrel{+}{=} K(x)$, we now can compute $\stackrel{+}{=} K(x)$.

ii) Follows from item i). □

Theorem 8.3: Given an oracle that on query x answers with a sufficient statistic f^* of x and a $c_x \stackrel{+}{=} 0$ as required below. Then, we can compute the Kolmogorov complexity function K and the halting sequence χ.

Proof: By Lemma 8.2, we can compute the function $K(x)$, up to fixed constant precision, given the oracle (without the value c_x) in the statement of the theorem. Let c_x in the statement of the theorem be the difference between the computed value and the actual value of $K(x)$. In [16, Exercise 2.2.7 on p. 175], it is shown that if we can solve the halting problem for plain Turing machines, then we can compute the (plain) Kolmogorov complexity, and *vice versa*. The same holds for the halting problem for prefix Turing machines and the prefix Turing complexity. This proves the theorem. □

Lemma 8.4: There is a constant c, such that for every x there is a program (possibly depending on x) of at most c bits that computes $\mathrm{soph}(x)$ and the witness program f from $x, K(x)$. That is, $K(f \mid x, K(x)) \stackrel{+}{=} 0$. With some abuse of notation we can express this as $K(\mathrm{soph} \mid K) \stackrel{+}{=} 0$.

Proof: By definition of sufficient statistic f^*, we have $K(f) + l_x(f) \stackrel{+}{=} K(x)$. By (V.2), the number of sufficient statistics for x is bounded by an independent constant, and we can generate all of them from x by a $\stackrel{+}{=} 0$ length program (possibly depending on x). Then, we can simply determine the least length of a sufficient statistic, which is $\mathrm{soph}(x)$. □

There is a subtlety here: Lemma 8.4 is nonuniform. While for every x we only require a fixed number of bits to compute the sophistication from $x, K(x)$, the result is nonuniform in the sense that these bits may depend on x. Given a program, how do we verify if it is the correct one? Trying all programs of length up to a known upper bound, we do not know if they halt or if they halt with the correct answer. The question arising is if there is a single program that computes the sophistication and its witness program for all x. In [21], this much more difficult question is answered in a strong negative sense: there is no algorithm that for every x, given $x, K(x)$, approximates the sophistication of x to within precision $l(x)/(10\log l(x))$.

Theorem 8.5: For every x of length n, and f^* the program that witnesses the sophistication of x, we have $K(f^* \mid x) \stackrel{+}{<} \log n$. For every length n, there are strings x of length n, such that

$$K(f^* \mid x) \stackrel{+}{>} \log n - \log\log n.$$

Proof: Let f^* witness the $\mathrm{soph}(x)$: That is, $K(f) + l_x(f) \stackrel{+}{=} K(x)$, and $l(f^*) = \mathrm{soph}(x)$. Using the conditional version of (II.4), see [6], we find that $K(K(x), f^* \mid x)$

$$\begin{aligned} &\stackrel{+}{=} K(K(x) \mid x) + K(f^* \mid K(x), K(K(x) \mid x), x) \\ &\stackrel{+}{=} K(f^* \mid x) + K(K(x) \mid f^*, K(f^* \mid x), x). \end{aligned}$$

In Lemma 8.2, item i), we show $K(K(x) \mid x, f^*) \stackrel{+}{=} 0$, hence also, $K(K(x) \mid f^*, K(f^* \mid x), x) \stackrel{+}{=} 0$. By Lemma 8.4, $K(f^* \mid K(x), x) \stackrel{+}{=} 0$, hence also $K(f^* \mid K(x), K(K(x) \mid x), x) \stackrel{+}{=} 0$. Substitution of the constant terms in the displayed equation shows

$$K(K(x), f^* \mid x) \stackrel{+}{=} K(f^* \mid x) \stackrel{+}{=} K(K(x) \mid x) \stackrel{+}{=} K(x^* \mid x). \tag{VIII.2}$$

This shows that the shortest program to retrieve f^* from x is essentially of the same length as the shortest program as to retrieve x^* from x or $K(x)$ from x. Using (VIII.1), this shows that

$$\log l(x) \stackrel{+}{>} \limsup_{l(x)\to\infty} K(f^* \mid x) \stackrel{+}{>} \log l(x) - \log\log l(x).$$

Since f^* is the witness program for $l(f^*) = \mathrm{soph}(x)$, we have $l(f^*) \stackrel{+}{=} K(f^*) \stackrel{+}{>} K(f^* \mid x)$. □

Definition 8.6: A function f from the rational numbers to the real numbers is *upper semicomputable* if there is a recursive function $H(x,t)$ such that $H(x,t+1) \leq H(x,t)$ and $\lim_{t\to\infty} H(x,t) = f(x)$. Here we interpret the total recursive function $H(\langle x,t\rangle) = \langle p,q\rangle$ as a function from pairs of natural numbers to the rationals: $H(x,t) = p/q$. If f is upper semicomputable, then $-f$ is *lower semicomputable*. If f is both upper-a and lower semicomputable, then it is *computable*.

Recursive functions are computable functions over the natural numbers. Since $K(\cdot)$ is upper semicomputable [16], and from $K(\cdot)$ we can compute $\mathrm{soph}(x)$, we have the following.

Lemma 8.7:

i) The function $\mathrm{soph}(x)$ is not computable to any significant precision.

ii) Given an initial segment of length $2^{2l(x)}$ of the halting sequence $\chi = \chi_1\chi_2\ldots$, we can compute $\mathrm{soph}(x)$ from x. That is,

$$K(\mathrm{soph}(x) \mid x, \chi_1 \ldots \chi_{2^{2l(x)}}) \stackrel{+}{=} 0.$$

Proof:

i) The fact that $\mathrm{soph}(x)$ is not computable to any significant precision is shown in [21].

ii) We can run $U(p,d)$ for all (program, argument) pairs such that $l(p)+l(d) \leq 2l(x)$. (Not $l(x)$ since we are dealing with self-delimiting programs.) If we know the initial segment of χ, as in the statement of the theorem, then we know which (program, argument) pairs halt, and we can simply compute the minimal value of $l(p)+l(d)$ for these pairs. □

IX. Discussion

"Sophistication" is the algorithmic version of "minimal sufficient statistic" for data x in the model class of total recursive functions. However, the full stochastic properties of the data can only be understood by considering the Kolmogorov structure function $\lambda_x(\alpha)$ (mentioned earlier) that gives the length of the shortest two-part code of x as a function of the maximal complexity α of the total function supplying the model part of the code. This function has value about $l(x)$ for α close to 0, is nonincreasing, and drops to the line $K(x)$ at complexity $\alpha_0 = \mathrm{soph}(x)$, after which it remains constant, $\lambda_x(\alpha) = K(x)$ for $\alpha \geq \alpha_0$, everything up to a logarithmic addive term. A comprehensive analysis, including many more algorithmic properties than are analyzed here, has been given in [21] for the model class of finite sets containing x, but it is shown there that all results extend to the model class of computable probability distributions and the model class of total recursive functions, up to an additive logarithmic term.

Acknowledgment

The author wishes to thank L. Antunes, L. Fortnow, K. Vereshchagin, and the referees for their comments.

References

[1] A. R. Barron, J. Rissanen, and B. Yu, "The minimum description length principle in coding and modeling," *IEEE Trans. Inf. Theory*, vol. 44, no. 6, pp. 2743–2760, Oct. 1998.

[2] T. M. Cover, "Kolmogorov complexity, data compression, and inference," in *The Impact of Processing Techniques on Communications*, J. K. Skwirzynski, Ed. Dordrecht, The Netherlands: Martinus Nijhoff , 1985, pp. 23–33.

[3] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.

[4] R. A. Fisher, "On the mathematical foundations of theoretical statistics," *Phil. Tran. Roy. Soc. London, Ser. A*, vol. 222, pp. 309–368, 1922.

[5] P. Gács, "On the symmetry of algorithmic information," *Sov.Math.—Dokl.*, vol. 15, pp. 1477–1480, 1974.

[6] P. Gács, J. Tromp, and P. Vitányi, "Algorithmic statistics," *IEEE Trans. Inf. Theory*, vol. 47, no. 6, pp. 2443–2463, Sep. 2001.

[7] Q. Gao, M. Li, and P. M. B. Vitányi, "Applying MDL to learn best model granularity," *Artificial Intell.*, vol. 121, pp. 1–29, 2000.

[8] M. Gell-Mann, *The Quark and the Jaguar*. New York: Freeman, 1994.

[9] P. D. Grünwald and P. M. B. Vitányi, Shannon Information and Kolmogorov Complexity. CWI, Amsterdam, The Netherlands, 2003, unpublished manuscript.

[10] A. N. Kolmogorov, "Three approaches to the quantitative definition of information," *Probl. Inf. Transm.*, vol. 1, no. 1, pp. 1–7, 1965.

[11] ——, "Complexity of algorithms and objective definition of randomness," presented at the Moscow Math. Soc. Meeting 4/16/1974, English translation in [21].

[12] ——, "On logical foundations of probability theory," in *Probability Theory and Mathematical Statistics (Lecture Notes in Mathematics)* K. Itô and Y. V. Prokhorov, Eds. Heidelberg, Germany: Springer-Verlag, 1983, vol. 1021, pp. 1–5.

[13] A. N. Kolmogorov and V. A. Uspensky, "Algorithms and randomness," *SIAM Theory Probab. Appl.*, vol. 32, no. 3, pp. 389–412, 1988.

[14] M. Koppel, "Complexity, depth, and sophistication," *Complex Syst.*, vol. 1, pp. 1087–1091, 1987.

[15] ——, "Structure," in *The Universal Turing Machine: A Half-Century Survey*R. Herken, Ed. Oxford, U.K.: Oxford Univ. Press, 1988, pp. 435–452.

[16] M. Li and P. Vitanyi, *An Introduction to Kolmogorov Complexity and Its Applications*, 2nd ed. New York: Springer-Verlag, 1997.

[17] C. E. Shannon, "The mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379–423, 623–656, 1948.

[18] ——, "Coding theorems for a discrete source with a fidelity criterion," in *IRE Nat. Conv. Rec., Pt. 4*, 1959, pp. 142–163.

[19] A. K. Shen, "The concept of (α,β)-stochasticity in the Kolmogorov sense, and its properties," *Sov. Math.—Dokl.*, vol. 28, no. 1, pp. 295–299, 1983.

[20] ——, "Discussion on Kolmogorov complexity and statistical analysis," *Comput. J.*, vol. 42, no. 4, pp. 340–342, 1999.

[21] N. K. Vereshchagin and P. M. B. Vitányi, "Kolmogorov's structure functions and model selection," *IEEE Trans. Inf. Theory*, vol. 50, no. 12, pp. 3265–3290, Dec. 2004.
[22] ——, "Arithmetic rate-distortion theory," *IEEE Trans. Inf. Theory*, submitted for publication.
[23] P. M. B. Vitányi and M. Li, "Minimum description length induction, Bayesianism, and Kolmogorov complexity," *IEEE Trans. Inf. Theory*, vol. 46, no. 2, pp. 446–464, Mar. 2000.
[24] V. V. V'yugin, "On the defect of randomness of a finite object with respect to measures with given complexity bounds," *SIAM Theory Probab. Appl.*, vol. 32, no. 3, pp. 508–512, 1987.
[25] ——, "Algorithmic complexity and stochastic properties of finite binary sequences," *Comput. J.*, vol. 42, no. 4, pp. 294–317, 1999.

Zero-Error Source–Channel Coding With Side Information

Jayanth Nayak, Ertem Tuncel, *Member, IEEE*, and Kenneth Rose, *Fellow, IEEE*

***Abstract*—This correspondence presents a novel application of the theta function defined by Lovász. The problem of coding for transmission of a source through a channel without error when the receiver has side information about the source is analyzed. Using properties of the Lovász theta function, it is shown that separate source and channel coding is asymptotically suboptimal in general. By contrast, in the case of vanishingly small probability of error, separate source and channel coding is known to be asymptotically optimal. For the zero-error case, it is further shown that the joint coding gain can in fact be unbounded. Since separate coding simplifies code design and use, conditions on sources and channels for the optimality of separate coding are also derived.**

***Index Terms*—Graph homomorphisms, Lovász theta function, source–channel separation, zero-error coding.**

I. Introduction

An information-theoretic result that has had a profound impact on practical communication system design is the separation theorem, which says that source and channel code design can be separated without any asymptotic loss of optimality. The first theorem of this kind was proved by Shannon [1] who considered the case where a discrete memoryless source needs to be communicated over a discrete memoryless channel and a nonzero reconstruction error that asymptotically vanishes as the code block length increases is allowed. This theorem has since been shown to hold for most analytically tractable single-user source–channel scenarios with a few exceptions under the asymptotically vanishing error constraint described previously [2]. Note that separation theorems are asymptotic results and make no claims about the behavior at finite block lengths.

A study of communication systems under the more stringent error-free constraint was also initiated by Shannon [3]. He characterized the zero-error capacity of the discrete memoryless channel both with and without feedback and established that the zero-error regime is different from the asymptotically vanishing error regime. For the source–channel pair of [1], the separation theorem trivially holds even under a zero-error constraint. The question of optimality of source–channel separation in the zero-error case becomes far more interesting when the decoder has access to side information about the source. For this communication scenario we resolve the question and demonstrate that zero-error behavior and the asymptotically vanishing error behavior differ substantially.

Let $\mathcal{C}$ be a discrete memoryless channel with transition probability $p_{Y|X}(y|x)$, $x \in \mathcal{X}$, $y \in \mathcal{Y}$, where $\mathcal{X}$ and $\mathcal{Y}$ are finite sets. With an asymptotically vanishing error requirement, the capacity of this channel is $C = \max_{p_X(x)} I(X;Y)$, where $I(X;Y)$ is the mutual information between X and Y. The zero-error capacity C_0, which was characterized by Shannon [3], will be discussed in detail in Section II.

Let $(\mathcal{S}_U, \mathcal{S}_V)$ be a pair of memoryless correlated sources producing realizations of a pair of random variables (U, V) from a finite set $\mathcal{U} \times \mathcal{V}$ at each instant. Alice, "the sender," has access to U while Bob, "the receiver," has access to V. Alice and Bob are connected by the channel $\mathcal{C}$. Alice employs (m, n) codes that map m realizations of U to n-length blocks of the channel input alphabet in order to noiselessly convey U. We wish to determine the minimum amount of channel resources required for Alice to convey U to Bob. We quantify the efficiency of a code by its rate $\frac{n}{m}$ channel uses per source symbol.

Suppose we wish to design a source–channel code for the source U with side information V and channel $\mathcal{C}$. The celebrated results of Shannon [1] and Slepian and Wolf [4] imply that communication is possible using separate source and channel codes if the rate is at least $\frac{H(U|V)}{C}$. On the other hand, Shamai and Verdú [5] have shown that codes with rate less than $\frac{H(U|V)}{C}$ cannot exist even if joint source–channel coding is employed. Hence, separate source and channel coding is asymptotically optimal when a vanishingly small probability of error is allowed.

In this correspondence, we focus on the *zero-error* setting for the problem of source–channel coding with side information. Section III presents our main results—the suboptimality of separate coding and the gains by joint coding. Our main tool in analyzing these problems is the theta function, a graph functional shown by Lovász to be an upper bound on the Shannon capacity of a graph [6]. Lovász employed the theta function to characterize the Shannon capacity of the pentagon graph, a problem that had remained open for more than two decades. To quantify the gains, we employ a graph construction by Alon that was used by him to disprove a conjecture of Shannon regarding the additivity of zero-error capacity with respect to channel sums [7]. In Section IV, we turn to the question of when separate coding is indeed optimal and present sufficient conditions on sources and channels. In Section V, we present some comments on the complexity of code design before concluding in Section VI. Since results in zero-error coding for the source and channel that we consider are not widely known, we first survey relevant aspects of this area in Section II.

II. Preliminaries and Notation

The imposition of zero-error constraints naturally leads to problem formulations in terms of graphs and we begin this section with some useful graph-theoretic definitions.

Manuscript received March 3, 2005; revised April 26, 2006. This work was supported in part by the National Science Foundation under Grant EIA-0080134, the University of California MICRO Program, Applied Signal Technology, Inc., Dolby Laboratories, Inc., Mindspeed Technologies, Inc., and Qualcomm, Inc. The material in this correspondence was presented in part at the IEEE International Symposium on Information Theory, Yokohama, Japan, June/July 2003.

J. Nayak is with INRIA/IRISA, Campus Universitaire de Beaulieu, 35042 Rennes, France (e-mail: jnayak@irisa.fr).

E. Tuncel is with the Department of Electrical Engineering, University of California, Riverside, CA 92521 USA (e-mail: ertem@ee.ucr.edu).

K. Rose is with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106 USA (e-mail: rose@ece.ucsb.edu).

Communicated by R. J. McEliece, Associate Editor for Coding Theory.

Digital Object Identifier 10.1109/TIT.2006.881718

0018-9448/$20.00 © 2006 IEEE