On Natural Representations of Objects

Fouad B. Chedid

Dhofar University, Salalah, Oman
fchedid@du.edu.om
and
Notre Dame University-Louaize, Zouk Mosbeh, Lebanon
fchedid@ndu.edu.lb

Abstract

We give explanations on the differences between a number of representations that have been proposed in the literature to measure the structural complexity of an object. In particular, we propose the notion of process complexity, the notion of sophistication from a given perspective, the notion of sophisticated depth, and the notion of self-dissimilarity. We also propose the notion of an approximate process for an object and argue that for all practical purposes, an approximate process is as good as an exact process.

1 Introduction

1.1 Related Work

Consider the process of theory formation in science. A scientist records a sequence of observations S about a given phenomenon P, and then based on S the scientist attempts to develop a theory that explains P. Let the sequence of observations S be encoded into a bit string x. First, we need to agree on what it is that we are trying to uncover about P. Are we after the true source of P? This seems to be a very hard task since we don't have the entire history (not to mention the future) of P and our explanation is to be based solely on a snapshot of the development of P as determined by x. So, our next best guess is to try to explain x, not P. Using the theory of algorithms, we can rely on the notion of universal Turing machine to define a complexity measure that is an intrinsic property of x itself (independent of any model). This is exactly what Solomonoff [19], Kolmogorov [11], and Chaitin [3] did in the 1960s (though each for his own reason and in his own way) when they defined the complexity $C(x)$ of a string x as the length of a shortest program p that generates x on a fixed reference universal Turing machine U. Formally,

$$C(x) = \min_p\{|p| : U(p) = x\}.$$

According to this complexity measure, named Kolmogorov complexity, it is the size of the program that generates x that determines the complexity of (amount of information in) x. In other words, strings that admit short descriptions have low complexity (little information) and strings that resist short descriptions have high complexity (lots of information). A simple counting argument shows that there exists one string of each length n that cannot be compressed at all (not by a single bit) and that more than half the strings of length n cannot be compressed by more than one bit. A string x for which $C(x) \geq |x|$ is said to be incompressible or algorithmically random (also called Kolmogorov random). A string x for which $C(x) \geq |x| - c$, for some constant c, is said to be c-incompressible. The prefix Kolmogorov complexity $K(x)$ of a finite string x is the length of a shortest prefix-free program to generate x on a universal prefix Turing machine. For a wealth of information on Kolmogorov complexity and its applications, the reader is referred to [14].

We next ask what quantity is really measured by $C(x)$? To motivate the discussion, consider, for example, the case of two strings x and y of 1000 characters each, where x is a passage from the book War and Peace and y is a string of 1000 characters chosen at random. Using the definition of Kolmogorov complexity, we conclude that y is more complex (has more information) than x. This is true because it is always possible to encode the string x by abbreviating some sentences in it to be recovered by the reader (say we use "btw" for "by the way", etc.). However, the only way to generate y is to copy it in its entirety. This confusion is due to the fact that $C(x)$ is really about the incompressibility of x, not the amount of meaningful information in x (what is meaningful?). Of course, we would all agree that y is more random than x. So it seems that $C(x)$ is better termed the amount of randomness in x rather than the amount of information. Now, what about the term complexity? It is hardly believable that a random string (y above) is more complex than a passage from Tolstoy's master piece (x above)! So, again the terminology "Kolmogorov Complexity" is a bit misleading and it would be more appropriate to call it "Kolmogorov Randomness". We mention that these ideas agree with the work of the German physicist Grassberger [9] and the American physicist Gell-Mann [8]. Grassberger suggests using the words "randomness" or "information contents" for the property measured by Kolmogorov complexity and reserving "complexity" for something that is between total regularity and total randomness. Thus, according to Grassberger, strings of total regularity or total randomness have small complexity. In either case, the string can be generated by a simple procedure. Here, simplicity is not related to the size of the procedure, but rather to the process involved in the generation of the string. Similarly, Gell-Mann supports the unsuitability of the quantity measured by Kolmogorov complexity as a measure of complexity. Furthermore, Gell-Mann suggests his own measure of complexity named effective complexity, which measures the information contents of the ensemble in which the object is embedded as a typical member.

In 1973, Kolmogorov (though he never published anything on this himself) proposed a structure function $h_x()$ (known as Kolmogorov's structure function) [5] as a way to divide a shortest program x^* for a finite binary string x into two parts responsible for encoding the regular and nonregular (random) aspects of x separately. Of course, both parts are maximally random (incompressible) and so it is not clear how such a division can be made. Still, the idea is that while we agree that $|x^*|$ is a measure of the amount of information (let's call it total information) in x, we would like to know more about the constituents of x^*. Mainly, how much of x^* is responsible for coding the regular aspects of x (to be associated with useful information in x) and how much of x^* is responsible for coding the random aspects of x (to be associated with accidental information in x). Kolmogorov [5] suggested to model the regular aspects of x by a finite set S of which x is a typical element. The notion of Kolmogorov's minimal sufficient statistic as the length of an optimally compressed description of the regular aspects of a string in the model class of finite sets was then coined. A string for which its minimal sufficient statistic is about $|x|$ was called absolutely nonrandom (nonstochastic) by Kolmogorov. Thus, an absolutely nonrandom string x contains almost no random information. Furthermore, almost all regularities in x are incompressible. Kolmogorov called these objects random in the negative sense. Interestingly, this incompressibility property of absolutely nonrandom objects is shared with Kolmogorov random objects. Random objects are incompressible because they contain a few regularities. Kolmogorov called these objects random in the positive sense. The following lemma by Gács *et al.* [7] answers a question raised by Kolmogorov about the existence of absolutely nonrandom strings.

Lemma 1. *For every length n, there exist strings x of length n with $K(x|n) = n + O(1)$ for which $\{x\}$ is an explicit minimal sufficient statistic.*

We mention that this lemma was first proved by Levin in 1973 (unpublished).

Another important measure for the regular aspects of a string in the model class of total recursive functions is provided by Koppel's notion of sophistication [12]. According to Koppel, the strings which are most complex are neither the structureless ones nor the ones with very simple structures. Rather, they are the ones with more sophisticated structures. Koppel defines the sophistication of an infinite string x as the size of the most concise description of the regular aspects of x based on monotonic complexity defined by Schnorr [17]. Behind this definition is the idea that a string with a sophisticated structure is part of a natural class of strings, all of which possess the same underlying structure [10]. Clearly, this idea draws on the suggestion of Kolmogorov of modeling the regular aspects of x by a finite set S of which x is a typical element.

Another important indirect measure for the amount of regular information in a string is provided by Bennett's notion of logical depth. According to Bennett [2], the value of a string is neither about the obviously observed nor the totally random parts in the string, it is more about the buried regularity part that can be uncovered but only after exerting a considerable amount of effort. To quantify this feature, Bennett defined the notion of logical depth. A string is called logically deep if it takes a lot of effort to generate it on a universal Turing machine using any short program for it. The logical depth $depth_s(x)$ of a string x, at significance level s, is defined as follows.

$$depth_s(x) = \min_p \{t : U^t(p) = x \text{ and } |p| \leq K(x) + s\}.$$

Here, p is a prefix-free program that generates x on a universal prefix Turing machine U that makes no more than t steps. The program p may contain up to s bits of buried regularity. The constant s can be viewed as the price that one has to pay in order to obtain a fast source for x like p. Observe that, by this definition, Kolmogorov random (in the positive sense) strings are logically shallow. Complex strings, on the other hand, must be logically deep.

Another important notion that shares much of its motivation with logical depth is thermodynamic depth [15]. Here, complexity is about how difficult it is to generate an object. Furthemore, complexity is not a property of a single physical state but rather a property of a process.

In 2006, Vitányi [21] developed the theory of recursive function statistic and explained the importance of using a total recursive function to model the regular aspects of a string if we are to be able to identify absolutely nonstochastic objects. This is because in the model class of partial recursive functions, all objects have low minimal sufficient statistic (the complexity of the model part can always be shifted to the data-to-model part of a universal Turing machine of constant complexity).

In 2009, Antunes and Fortnow [1] adapted Koppel's notion of sophistication for finite strings based on prefix Kolmogorov complexity. The new definition is as follows. The c-sophistication of a finite binary string x is [1]:

$$soph_c(x) = \min_{\text{total } p} \{|p| : \exists d, U(p, d) = x \text{ and } |p| + |d| \leq K(x) + c\}.$$

Here, U is a fixed reference universal prefix Turing machine, p is the self-delimiting encoding of a halting prefix Turing machine T that summarizes all regularities in x, the string d is a program that when interpreted by T gives x, and the term c is a small constant used to indicate how far is $|p| + |d|$ from the length of a shortest program for x. Thus, strings which are Kolmogorov random in the positive sense have minimum sophistication. On the other hand, strings which are Kolmogorov random in the negative sense must be highly sophisticated.

Finally, several notions of time-bounded Kolmogorov complexity have been proposed. One of the most useful one is by Levin[13]. Let U be a universal Turing machine. Define

$$Kt(x) = \min_p\{|p| + \log t : U(p) = x \text{ in at most } t \text{ steps.}\}$$

Clearly, $\log |x| \leq Kt(x) \leq |x| + O(\log |x|)$.

1.2 This Work

This work is inspired by the work of Goertzel in [10]. We use many of the ideas discussed in [10] to propose new notions of complexity and give explanations on the differences between the various representations reviewed in this section. In particular, we propose the notion of process complexity, the notion of an approximate process, the notion of sophistication from a given perspective, the notion of sophisticated depth, and the notion of self-dissimilarity.

2 Introducing the Notion of Process Complexity

It is known that Kolmogorov complexity is in general incomputable (a consequence of the undecidability of the halting problem). In this section, we introduce a computable variant of Kolmogorov complexity named Process Complexity. We achieve computability by bounding the running time of a universal Turing machine.

Let $\Sigma = \{0, 1\}$. Let x be a finite string over Σ (i.e., $x \in \Sigma^*$). Then, we have the following definition for a process for x:

Definition 2 (Process Definition)**.** *Let U be a universal Turing machine. Then, a process for a finite binary string x is an ordered triple (p, d, t), where p is a program, d is an input and $U(p, d)$ halts in at most t steps and outputs x.*

We think of the program component p as the part of the process that summarizes the structure of the objects generated by that process. We think of the process components d and t as the parts that represent the amounts of input and effort needed to generate a specific instance from the set of objects generated by the process, respectively. With this in mind, we make the following two observations:

1. Consider processes having different program components. If these processes generate the same object, then that object must have different levels of structure. We use this idea to propose in Section 3 the notion of sophistication from a given perspective.

2. Consider processes having the same program component but different data components. Then, these processes can only generate objects having the same underlying structure. We use this idea to propose in Section 5 the notion of an approximate process for an object. Basically, an approximate process for an object x doesn't generate x exactly but a variation of x belonging to the same natural class containing x.

Let the length of a process (p, d, t) be $|p| + |d| + t$. A straighforward process for any string x is the "print x" process whose description has length $|p| + |d| + t = 2|x| + c$, where c is a positive integer. This is true because the "print x" process runs the constant-length program "print" on the data x of length $|x|$ in $|x|$ time steps.

Following [10], we propose the following definition:

Definition 3 (Process Efficacy)**.** *The efficacy of a process (p, d, t) is the percentage of savings in process description obtained from using the process (p, d, t) for x over the print process $(print, x, |x|)$. In the following, we neglect the length c of the copy program.*

$$\mathit{Eff}((p, d, t), x) = \frac{2|x| - (|p| + |d| + t)}{2|x|} = 1 - \frac{|p| + |d| + t}{2|x|} \leq 1.$$

Thus, an efficacy that is close to 1 indicates a very efficient process. In this case, the process (p, d, t) uncovers lots of regularities in x in a clever way. On the other hand, if the efficacy is close to zero, then the process (p, d, t) doesn't improve much on the simple copy process, a case which could be attributed to the lack of regularities in x. Thus, incompressible strings cannot have efficient processes.

The following definition describes the most efficient process for a string:

Definition 4 (Process Complexity)**.** *Let U be a universal Turing machine. Then, the process complexity of a finite binary string x is*

$$PC(x) = \min_{(p,d,t)} \{|p| + |d| + t : (p, d, t) \text{ is a process for } x\}.$$

Thus, the process complexity of a string is the length of the most concise (space and time wise) process for that string.

We also have the following:

Definition 5 (Randomness)**.** *A string is said to be random if its process complexity is about the length of its copy process. Formally put, a string x is called c-random (where $c \in \mathbb{N}$) if $PC(x) \geq 2|x| - c$.*

3 Process Crudity

Assume that a string x can be generated by two different processes (p_1, d_1, t_1) and (p_2, d_2, t_2). What kinds of questions can one ask about these processes? First, we can use the efficacy function Eff() to determine which process provides a shorter description. However, what is even more important about a process is the amount of regularities uncovered by it. By this we mean the size of the program component of the process. Then, in choosing between different processes for the same object x, one should seek a process (p, d, t) for x whose progam size $(|p| + |d|)$ is small, whose running time t is small, and whose uncovered regularities $|p|$ is large (the best of such processes is the most concise, most efficient, and most sophisticated process for x). Notice that these three criteria can be satisfied by minimizing the following quantity [10]:

$$\frac{|p| + |d| + t}{|p|}$$

which is equivalent to minimizing

$$\frac{|d| + t}{|p|}.$$

Following Goertzel [10], we call this quantity the crudity of a process. Thus, a best process for an object is a process with the least crudity. Observe that crudity is a much stronger notion than process efficacy. This is true because while crudity implies efficacy (a least crude process

is a most efficient process), the reverse is not necessarily true (a most efficient process may not be a least crude proccess).

The following is a complexity measure that describes the least crude process for a string.

Definition 6 (Process Crudity Complexity)**.** *Let U be a universal Turing machine. Then, the process crudity complexity of a finite binary string x is*

$$CC(x) = \min_{(p,d,t)} \left\{ \frac{|d| + t}{|p|} : (p, d, t) \text{ is a process for } x \right\}.$$

We label the program p associated with a least crude process for x as a most natural program for x and the length of the program p as a number specifying a most natural class $S_{|p|}$ containing x.

Next, suppose that there are several least crude processes (p_i, d_i, t_i) (with different program components p_i) for the same object x. Then, these processes must be generating x from different perspectives. In other words, the object x must have many levels of order. We hypothesize that such a condition is sufficient for x to be complex. This hypothesis agrees with the work of Crutchfield and Young in [6] and Simon's famous paper on the architecture of complexity [18]. According to [6], "The idea is that a data set is complex if it is the composite of many symmetries." According to [18], in complex systems there are often many levels of organization that can be thought of as forming a hierarchy of systems and sub-systems. Observe that randomness is no longer an important factor here.

We next consider the total sophistication of an object x that can be generated from different perspectives. Clearly, that value must combine the sophistications of x from various perspectives. In the following, we introduce our notion of self-dissimilarity which allows sophistication to be decomposible in a linear way. We mention that the term self-dissimilarity was first coined in [23] where it was used in an argument that is totally different from what we have here.

Definition 7. *An object is said to be* self-dissimilar *for a number of perspectives if the regularities exhibited over those perspectives are independent.*

We have the following for a self-dissimilar object x:

$$\text{total-soph}(x) = \sum_i |p_i|,$$

where p_i is the program component of a least crude process for x. We mention that in this (simpler) case, the notion of total sophistication conforms with our intuitive notion of a quantity.

We have the following theorem:

Theorem 8. *A finite string x is absolutely nonstochastic if it can be observed from about $|x|$ perspectives for which x is self-dissimilar.*

Proof. If x is self-dissimilar for about $|x|$ perspectives, then it follows that total-soph$(x) = \sum_{i=1}^{|x|} |p_i|$, where p_i is the program component of a least crude process for x. Thus, total-soph$(x) \geq |x|$ and hence x is trivially absolutely non-stochastic. □

4 Crudity and Randomness

In this section, we use the notion of process crudity to redefine randomness.

Definition 9. *A string x is said to be random if it has a maximal crudity complexity. Thus, for a random string x, all processes (p, d, t) for x have either large $\frac{|d|+t}{|p|}$ or large $\frac{|d|}{|p|}$ or large $\frac{t}{p}$.*

We shall see that defining randomness this way gives good explanations on the difference between Kolmogorov's random objects in the positive sense and logically deep objects. In particular, we consider the following two cases:

Case of large $\frac{|d|}{|p|}$. Here, the size of the random part d of any least crude process for x is relatively much larger than the size of the structural part p of that process. This makes x Kolmogorov random in the positive sense but not necessarily logically deep. We think of the ratio $\frac{|d|}{|p|}$ as a measure of the ad-hocness of the process. Thus, processes with maximal (minimal) ad-hocness are candidates for generating Kolmogorov random objects in the positive (negative) sense.

Case of large $\frac{t}{|p|}$. Here, the amount of time t needed to generate x by any least crude process for x is relatively much larger than the size of the structural part p of that process. This makes x random and logically deep. We use this argument to propose a variant of logical depth named sophisticated depth, soph-depth$_s$(), as follows.

$$\text{soph-depth}_s(x) = \min_p \left\{ \frac{t}{|p|} : \exists d, U^t(p, d) = x \text{ and } |p| + |d| \leq K(x) + s \right\}.$$

Thus, while the logical depth of Bennett is a measure of the running time of a very concise program for an object, our notion of sophisticated depth is about the running time of a very concise and very sophisticated program for x. Notice that soph-depth() is a much stronger notion than *depth*(). This is because while soph-depth() implies *depth*(), the reverse is not necessarily true.

5 Approximate Process

We say that an ordered triple (p, d, t) is an approximate process for a string x if that process doesn't generate x exactly but a variation x_s of x such that x and x_s have the same underlying structure (x_s belongs to the same natural class containing x). Thus, processes for x and x_s will have the same program component p but different data components, say d and d_s. In this sense, an approximate process for x is, for all practical purposes, as good as a process for x for it preserves x's structural complexity. To quote Wallace's slogan "A tiger is any pattern which behaves as a tiger" [22].

We formally define what it means for a string x_s to be an approximation of a string x and what it means for a process to be an approximate process for a string.

Definition 10. *Let x and x_s be two finite strings from the same alphabet. Then, x_s is said to be an approximation of x if there are least crude processes (p, d, t) and (p_s, d_s, t_s) for x and x_s, respectively, such that $p = p_s$. The process (p, d_s, t_s) will be called an approximate process for x.*

6 Acknowledgments

The author wishes to acknowledge the anonymous reviewer #1 for his/her detailed and helpful comments.

References

[1] Luis Antunes and Lance Fortnow. Sophistication revisited. *Theory of Computing Systems*, 45(1):150-161, 2009.

[2] Charles Bennett. Logical depth and physical complexity. In R. Herken, editor, *The Universal Turing Machine: A Half-Century Survey*, pages 227-257, Oxford University Press, 1988.

[3] Gregory Chaitin. On the length of programs for computing finite binary sequences. *Journal of theACM* 13(4):145-149, 1966.

[4] Thomas Cover and Joy Thomas. *Elements of Information Theory*. Wiley Series in Telecommun. New Yoirk: Wiley, 1991.

[5] Thomas Cover. Kolmogorov complexity, data compression, and inference. In J. K. Skwirzynski, editor, *The Impact of Processing Techniques on Communications*, pages 23-33, Martinus Nijhoff Publishers, 1985.

[6] Jim Crutchfield and Karl Young. Computation at the onset of chaos. In Wojciech H. Zurek, editor, *Complexity, Entropy and the PHysics of Information*, Vol. 8, pages 223-269, The Santa Fe Institute, 1990.

[7] Peter Gács, J. Tromp, and P. Vitányi. Algorithmic statistics. *IEEE Trans. on Information Theory* 47(6):2443-2463, 2001.

[8] Murray Gell-Mann and Seth Lloyd. *Effective Complexity*, pages 387-398, The Santa Fe Institute, 2003.

[9] Peter Grassberger. Information and complexity measures in dynamical systems. *Information Dynamics*, Plenum Press, 1990.

[10] Ben Goertzel. *The structure of Intelligence: A New Mathematical Model of Mind*. Springer, 1993.

[11] Andrey Kolmogorov. Three approaches to the quantitative definition of information. *Problems Inform. Transmission* 1(1):1-7, 1965.

[12] Moshe Koppel. Complexity, depth, and sophistication. *Complex Systems* 1:1087-1091, 1987.

[13] Leonid Levin. Randomness conservation inequalities; information and independence in mathematical theories. *Information and Control*, 61:15-37, 1984.

[14] Ming Li and Paul Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications* Third Edition. Springer-Verlag, 2008.

[15] Seth Lloyd and Heinz Pagels. Complexity as thermodynamic depth. *Annals of Physics* 188(1):186-213, 1988.

[16] Nihat Ay, Markus Muller, and Arleta Szkota. Effective complexity and its relation to logical depth. ' *IEEE Trans. on Information Theory* 56(9):4593-4607, 2010.

[17] Claus Schnorr. Process complexity and effective random tests. *J. Comp. System Sci.* 7:376-388, 1973.

[18] Herbert Simon. The architecture of complexity. " *Proc. of the American Philosophical Society* 106(6):467-482, 1962.

[19] Ray Solomonoff. A formal theory of inductive inference, Part I. *Information and Control* 7(1):1-22, 1964.

[20] Nikolay Vereshchagin. Algorithmic minimal sufficient statistics: a new definition. *ECCC* TR10-090, 2010.

[21] Paul Vitányi. Meaningful information. *IEEE Transactions on Information Theory* 52(10):4617-4626, 2006.

[22] David Wallace. Everett and structure. *Studies in History and Philosophy of Modern Physics* 34(1):87-105, 2003.

[23] David Wolpert and William Macready. Self-dissimilarity: an empirically observable measure of Complexity. *Proc. of the First NECSI International Conference*, pp. 626-643, 2002.