Complex Systems **1** (1987) 1087–1091

Complexity, Depth, and Sophistication

Moshe Koppel
Queens College, Flushing, NY 11367, USA

Abstract. Two measures of the "meaningful" complexity of an infinite string are shown to be equivalent up to a constant (under appropriate translation). "Sophistication", defined by Koppel and Atlan [3], is the size of the projectible part of the string's minimal description and formalizes the amount of planning which went into the construction of the string. "Depth", defined by Bennett [1], is the amount of time required for the string to be generated from its minimzl description and formalizes its "evolvedness."

1. Introduction

The total complexity of an object is defined as the size of its most concise description. The total complexity of an object can be large while its "meaningful" complexity is low; for example, a random object is by definition maximally complex but completely lacking in structure. In the analysis of complex systems, we are more often interested in the growth of meaningful complexity than in the growth of total complexity. Thus, it is important that the former be formally defined. Meaningful complexity can be formalized in terms of static resources as the amount of planning which likely went into the construction of an object or it can be measured in terms of dynamic resources as the amount of computation needed to execute this plan. In this paper, we will unify these two approaches.

The "static" approach to the formalization of meaningful complexity is "sophistication" defined and discussed by Koppel and Atlan [3]. Sophistication is a generalization of the "H-function" or "minimal sufficient statistic" of Cover and Kolmogoroff [2], using the motonic complexity of Levin [4]. The sophistication of an object is the size of that part of the most concise description of that object which describes its structure, i.e. the aggregate of its projectible properties. For example, the sophistication of a string which is random except that each bit is doubled (e.g. 00110000110011...) is the size of the part of the description which represents the doubling of the bits.

The "dynamic" approach to the formalization of meaningful complexity is "depth" defined and discussed by Bennett [1]. The depth of an object is the running-time of its most concise description. Since it is reasonable to assume that an object has been generated by its most concise description, the depth of an object can be thought of as a measure of its evolvedness.

Although sophistication is measured in integers and depth is measured in functions, it is not difficult to translate to a common range. It has already been shown by Schnorr and Fuchs [5] that the sophistication of an infinite string is infinite if and only if its depth is infinite. In this paper, we will prove that for all infinite strings sophistication and depth are essentially equal (that is, they differ by at most some constant). Thus, *the more sophisticated an object the more time needed for its evolution.*

One way of demonstrating the naturalness of a concept is by proving the equivalence of a variety of *prime facie* different formalizations (e.g. computability). It is hoped that the proof of the equivalence of two approaches to meaningful complexity, one using static resources (program size) and the other using dynamic resources (time), will demonstrate not only the naturalness of the concept but also the correctness of the specifications used in each formalization to ensure robustness and generality.

2. Complexity

Let U consist of four tapes: a program tape, a data tape, a work tape, and an output tape. Given the contents of the cells being scanned in some state, U can switch states, move one of the scanners left or right or print a 0 or 1 in one of the cells being scanned. These instructions are restricted as follows:

1. The program, data, and output tapes are scanned left to right only.
2. U writes on the output tape only if the cell scanned is blank and moves right on the output only if the cell scanned is not blank.
3. The computation halts if and only if a blank is scanned on the data tape.

If beginning in some fixed initial state, with the finite binary string P on the program tape and the (possibly infinite) binary string D on the data tape, and the program and data scanner on the first bits of P and D, respectively, the computation halts with the (possibly empty) binary string S on the output tape, then we say that $U(P, D) = S$. If the computation does not halt but continues printing bits of the infinite binary string α on the output tape then we say that $U(P, D) = \alpha$. Let S^n be the initial segment of length n of the string S. A *process* is a function f from finite binary strings to (finite or infinite) binary strings such that for all S, n we have $f(S^n)$ is an initial segment of $f(S^{n+1})$ (if $f(S^n)$ is an infinite string, then $f(S^{n+1}) = f(S^n)$).

Clearly, a function f for which there exists P such that $U(P, D) = f(D)$ is a process. If for every partially computable process F there exists a program P such that $U(P, D) = F(D)$ for all D, then we call U a Universal Turing Machine.

The significance of processes as minimal descriptions is also discussed in [4].

A program P is *total* if $U(P, D)$ is defined for all D. Note that $U(P, D)$ can be finite or infinite.

A program is *self-delimiting* if during the course of the computation of $U(P, O)$ the program scanner reads the last bit of P but does not go beyond it.

Definition 1. *The* complexity *of* $S, H(S) = \min\{|P| + |D| \,|P$ *is a total and self-delimiting and* $U(P, D) \supseteq S\}$.

3. Sophistication

Definition 2. *A description of* α, (P, D), *is* c-minimal *if* $|P| + |D| \leq H(\alpha) + c$.

Definition 3. *The* c-sophistication *of a finite string* S, $SOPH_c(S) = \min\{|P| \,|\exists D$ *s.t.* (P, D) *is a c-minimal description of* $\alpha\}$.

Definition 4. *The* weak sophistication *of an infinite string* α, $SOPH'_\alpha = \min\{|P| \,|\exists\, c \forall n \;\exists\, D_n (P, D)$ *is a c-minimal description of* $\alpha\}$.

Definition 5. *The* (strict) sophistication *of an infinite string* α, $SOPH(\alpha) = \min\{|P| \,|\exists\, c \forall\, n \exists\, D_n\; (P, D_n)$ *is a c-minimal description of* α *and* $D_n \subseteq D_{n-1}\}$.

Definition 6. *An infinite string* α *is* transcendent *if* $\forall c \;\overline{\lim}\; SOPH_c(\alpha^n) = \infty$. *If* α *is transcendent, we define* $SOPH'(\alpha) = SOPH(\alpha) = \infty$.

Observe that it is not immediate from the definition that SOPH′ and SOPH are defined for all α. We will prove, however, that SOPH′ is defined for all α.

4. Depth

Every finite or infinite string β has at most one initial segment which is a self-delimiting program in U. If such an initial segment of β exists, call it β^P and call the rest of β, β^D.

Let RUN be a program in U such that for all β for which β^P is defined, $U(\mathrm{RUN}, \beta) = U(\beta^P, \beta^D)$.

For a strictly increasing function $F : N \to N$, let $U^F(P, D) =$ the resulting of aborting the computation of $U(P, D)$ if $F(n)$ steps have been executed and less than n bits of output have been printed; after aborting, 0's are printed on the output tape forever.

For a strictly increasing function F, let γ_F be the characteristic string of the range of F and let $|F| = \min\{|P| \,|P$ is a self-delimiting program and $U(P, O) = \gamma_F\}$.

Definition 7. *The weak depth of an infinite string α, $D'(\alpha) = \min\{|F| \mid \exists\ c \forall\ n \exists\ \beta_n\ (\beta_n^P, \beta_n^D)$ is a c-minimal description of α^n and $U^F(RUN, \beta_n) = U(RUN, \beta_n)\}$.*

If no such F exists, then $D'(\alpha) = \infty$.

Definition 8. *The (strict) depth of an infinite string α, $D(\alpha) = \min\{|F| \mid \exists\ c \forall\ n \exists\ \beta_n\ (\beta_n^P, \beta_n^D)$ is a c-minimal description of α^n and $\beta_n \supseteq \beta_{n-1}$ and $U^F(RUN, \beta_n) = U(RUN, \beta_n)\}$.*

If no such F exists then $D(\alpha) = \infty$.

Thus, the depth of an infinite string α is the size of the smallest program which computes an upper-bound on the running-time of minimal descriptions of segments of α. (The reader should take note of the hidden role played here by a "busy beaver"-type function.)

5. Equivalence of sophistication and depth

Theorem 1. *SOPH'(α) is defined for all α. Moreover, there exists c such that for all α, either SOPH'$(\alpha) = D'(\alpha) = \infty$ or $|SOPH'(\alpha) - D'(\alpha)| < c$.*

Proof. We first show that if $D'(\alpha) = \infty$ then $\mathrm{SOPH}'(\alpha) = \infty$. If SOPH $\neq \infty$, then for some c and all n there exists ℓ such that $\mathrm{SOPH}_c(\alpha^n) \leq \ell$. But then the programs whose lengths determine the c-sophistications of the various α^n, come from among a finite set of total, self-delmiting programs, $\{P_i\}$, $1 \leq i \leq k$, $k \leq 2^\ell$. For any total program P, let $ST_P(m)$ be the maximum number of steps which $U(\mathrm{RUN}, P \cdot D)$ runs before either halting or producing m bits of output, with the maximum taken over all D. Then, $ST_P(m)$ is a total, recursive function. Let $g(m) = \max\{ST_{P_i}(m)\}$. Then, choosing β_n such that $\beta_n^P \in \{P_i\}$, $1 \leq i \leq k$ and $1 \leq i \leq k$ (β_n^P, β_n^D) is a c-minimal description of α^n, we have $U^g(\mathrm{RUN}, \beta_n) = U(\mathrm{RUN}, \beta_n)$ and thus $D'(\alpha) \leq |g| < \infty$ and the claim follows.

Now we will show that for some c if $D'(\alpha)$ is finite then $|\mathrm{SOPH}'(\alpha) - D'(\alpha)| \leq c$. If $D'(\alpha)$ is finite, then there exists some recursive function F and some sequence of strings $\{\beta_n\}$ and some c' such that for all n, (β_n^P, β_n^D) is a c'-minimal description of α^n and $U^F(\mathrm{RUN}, \beta_n) = U(\mathrm{RUN}, \beta_n)$. Now let $\mathrm{RUN} \cdot F$ be a self-delimiting program such that $U(\mathrm{RUN} \cdot F, \beta) = U^F(\mathrm{RUN}, \beta)$. Then, for all n, $U(\mathrm{RUN} \cdot F, \beta_n)$ is a c-minimal description of α^n), so it follows that $\mathrm{SOPH}'(\alpha) \leq |\mathrm{RUN} \cdot F| \leq |F| + c = D'(\alpha) + c$.

Since $D'(\alpha)$ is defined for all α and we have shown that if $D'(\alpha) = \alpha$ then $\mathrm{SOPH}'(\alpha) = \infty$ and if $D'(\alpha)$ is finite then $\mathrm{SOPH}'(\alpha)$ is finite, it is proved that $\mathrm{SOPH}'(\alpha)$ is defined for all α.

It remains only to show that for some c, $D'(\alpha) \leq \mathrm{SOPH}'(\alpha) + c$. Suppose that $\mathrm{SOPH}'(\alpha)$ is determined by the length of the program P. Then, there exists ${}_{i=1}^{\infty}\{D_n\}$ and c' such that for all n, (P, D_n) is a c'-minimal description of α^n. Now letting $F = ST_P$ we have $U^F(\mathrm{RUN}, P \cdot D_n) = U(\mathrm{RUN}, P \cdot D_n) = U(P, D_n) = \alpha^n$. Thus, $D'(\alpha) = |F| = |ST_P| \leq |P| + c' = \mathrm{SOPH}'(\alpha) + c$, and the theorem is proved. ∎

Definition 9. *An infinite string α is* describable *if $SOPH(\alpha)$ is finite.*

Theorem 2. *There exists c such that for all describable α, $|SOPH(\alpha) - D(\alpha)| \le c$.*

The proof of this theorem is identical to the second part of the above proof with the consistent application of the restriction on $\{D_n\}$ and $\{\beta_n\}$.

References

[1] C. Bennett, "On the logical 'depth' of sequences and their reducibilities to incompressible sequences," to appear.

[2] T. Cover, "Kolmogorov complexity, data compression, and inference," in *The Impact of Processing Techniques on Communications*, J. K. Skwirzynski ed. (Martinus Nijhoff Publishers, 1985).

[3] M. Koppel and H. Atlan, "Program-length complexity, sophistication, and induction," submitted.

[4] L. A. Levin, "On the notion of a random sequence," *Soviet Math. Dokl.*, **14/5** (1973) 1413–1416.

[5] C. P. Schnorr and P. Fuchs, "General random sequences and learnable sequences," *J. Symb. Logic*, **42** (1977) 329–340.