CHICAGO JOURNAL OF THEORETICAL COMPUTER SCIENCE 2010, Article 4, pages 1–13
http://cjtcs.cs.uchicago.edu/

SPECIAL ISSUE FOR CATS 2009

On Process Complexity

Adam R. Day

Received: January 14, 2009; revised: September 23, 2009; published: June 22, 2010.

Abstract: Process complexity is one of the basic variants of Kolmogorov complexity. Unlike plain Kolmogorov complexity, process complexity provides a simple characterization of randomness for real numbers in terms of initial segment complexity. Process complexity was first developed in [11]. Schnorr's definition of a process, while simple, can be difficult to work with. In many situations, a preferable definition of a process is that given in [9]. In this paper we define a variant of process complexity based on Levin and Zvonkin's definition of a process. We call this variant strict process complexity. Strict process complexity retains the main desirable properties of process complexity. Particularly, it provides simple characterizations of Martin-Löf random real numbers, and of computable real numbers. However, we will prove that strict process complexity does not agree within an additive constant with Schnorr's original process complexity.

One of the basic properties of prefix-free complexity is that it is subadditive. Subadditive means that there is some constant d such that for all strings σ, τ the complexity of $\sigma\tau$ (σ and τ concatenated) is less than or equal to the sum of the complexities of σ and τ plus d. A fundamental question about any complexity measure is whether or not it is subadditive. In this paper we resolve this question for process complexity by proving that neither of these process complexities are subadditive.

1 Introduction

The basic concept behind Algorithmic Information Theory is that the inherent complexity of a string can be quantified by the minimum number of bits of information needed to describe it. This concept was first formalized by defining the complexity of a string σ, as the length of the shortest description of σ with respect to some universal interpreter. This is known as the plain Kolmogorov complexity of σ. However, plain Kolmogorov complexity, while useful for many purposes, does not truly capture this basic concept

© 2010 Adam R. Day
Licensed under a Creative Commons Attribution License
DOI: 10.4086/cjtcs.2010.004

[4, 10]. The problem is that the universal interpreter can use not only the bits within the description, but also the *length* of the description to determine its output.

There are several variants of Kolmogorov complexity that attempt to overcome this problem. Each variant provides different insight into the nature of complexity. These variants include the widely used prefix-free complexity as well as the process complexity introduced in [11]. One reason prefix-free complexity has been extensively studied is that the Kraft-Chaitin theorem and the coding theorem provide an easy means of constructing prefix-free machines. No similar theorems exist for process complexity and this complexity seems genuinely more difficult to deal with.

In order to give a formal definition of the complexities we will study, first we will define complexity with respect to a general function. Take any $F : 2^{<\omega} \to 2^{<\omega}$ (where $2^{<\omega}$ is the set of all finite binary strings). The complexity of the string σ with respect to F is:

$$C^F(\sigma) = \begin{cases} \min\{|\tau| : F(\tau) = \sigma\} & \text{if } \exists \tau \in 2^{<\omega}, \\ & F(\tau) = \sigma \\ \infty & \text{otherwise} \end{cases}$$

In the above definition, $|\tau|$ refers to the length of the string τ. We can define a universal Turing machine U by setting $U(0^e 1\tau) = M_e(\tau)$ where M_e is the eth Turing machine. Note that this definition gives us that for any Turing machine M, there is a constant c, such that $C^U(\sigma) \leq C^M(\sigma) + c$. A machine with this property is called *optimal*. In this paper we will require all our universal machines to be optimal.

We define the *plain Kolmogorov complexity* of a string σ to be $C(\sigma) = C^U(\sigma)$ where U is a fixed universal Turing machine. We call a finite string σ *random* if $C(\sigma) \geq |\sigma|$. Note that U is a universal machine for the *whole* class of Turing machines. One way to produce variants of Kolmogorov complexity is to use a universal machine for a subclass of Turing machines. Two important subclasses of Turing machines are the class of all prefix-free machines and the class of all process machines. To understand these machines, first note that $2^{<\omega}$ has the following natural partial ordering. If $\tau_1, \tau_2 \in 2^{<\omega}$ then we define $\tau_1 \leq \tau_2$ to hold if τ_2 is an extension of τ_1. A subset A of $2^{<\omega}$ is called *prefix-free* if it is anti-chain with respect to this partial ordering, or alternatively if for all distinct τ_1, τ_2 in A, $\tau_1 \not\leq \tau_2$.

A *prefix-free machine* is a partial computable function $2^{<\omega} \to 2^{<\omega}$ whose domain is a prefix-free set. It can be shown that there exists a universal prefix-free machine and by fixing U this time as a universal prefix-free machine we can similarly define the prefix-free complexity of σ by $K(\sigma) = C^U(\sigma)$.

Another natural subclass is the class of all process machines. The key idea about a process is that it preserves the ordering on $2^{<\omega}$ making its action continuous.

Definition 1.1. A *process machine* is a partial computable function $M : 2^{<\omega} \to 2^{<\omega}$ such that if $\tau, \tau' \in \mathrm{dom}(M)$, and $\tau' \leq \tau$, then $M(\tau') \leq M(\tau)$.

A natural example of a process is given by the recordings of moves in a game of chess e.g. 1. e4 e5, 2. Nf3 Nc6 etc. Given the descriptions of the first n moves, it is possible to replay the game up until that point. If we extend the description, we can extend the replay of the game by adding new moves, but we cannot change any moves already defined.

Again we can take a universal process machine U and define the *process complexity* K_{M_D} of a string σ by $K_{M_D}(\sigma) = C^U(\sigma)$. This definition of a process machine and related complexity was given in [11]. We

follow the notation of [4] by using K_{M_D} to denote process complexity. Schnorr's definition of a process differs slightly from that given in [9]. We will use the term strict process machine for Levin and Zvonkin's definition. It is defined as follows.

Definition 1.2. A *strict process machine* is a partial computable function $M : 2^{<\omega} \to 2^{<\omega}$ such that if $\tau \in \mathrm{dom}(M)$ and $\tau' \leq \tau$, then $\tau' \in \mathrm{dom}(M)$ and $M(\tau') \leq M(\tau)$.

While Levin and Zvonkin defined strict process machines, they did not use them to define a measure of complexity in the same way as Schnorr. Instead they used strict process machines to construct a universal semimeasure from which another variant of Kolmogorov complexity was defined [9, 7].

Both of these definitions of process machines have merit. Schnorr's definition corresponds to a homomorphism of the domain of M. Levin and Zvonkin's definition has the following very natural model. This model is almost identical to one described in the first paper on algorithmic randomness [13]. Take a three-tape Turing machine M with a read-only one-way input tape, a one-way write-once output tape, and a work tape. The first square of the input table is blank and the input head starts on that square. Let the machine run. If at any stage M wants to move the input head of the tape, first we define $M(\tau) = \sigma$, where τ is the input string read so far and σ is the current output on the output tape.

Levin and Zvonkin's definition of a process can be easier to deal with. This becomes apparent when attempting any game based proof where the opponent is developing a process machine. If the opponent must keep the domain of the process machine closed under substrings, then the game becomes much simpler. For example, a game based technique can be used to show that there exists a universal strict process machine whose complexity overgraph (the set of pairs (σ, n) such that the complexity of σ is less than or equal to n) is not truth-table complete [3]. However, this technique cannot be extended to process machines in general and it is unknown if the result holds in this case.

Definition 1.3. Given any $\sigma \in 2^{<\omega}$, the *strict process complexity* $K_{M_S}(\sigma)$ is defined by $C^U(\sigma)$ where U is a universal strict process machine.

The existence of a universal strict process machine is established by [9]. Note that a prefix-free machine can be considered as a strict process machine. This is done as follows. Suppose $M : 2^{<\omega} \to 2^{<\omega}$ is a prefix-free machine. We take λ to be the empty string and we define a strict process machine M' as follows:

$$M'(\sigma) = \begin{cases} M(\sigma) & \text{if } \sigma \in \mathrm{dom}(M) \\ \lambda & \text{if there exists } \sigma' > \sigma \text{ and} \\ & \sigma' \in \mathrm{dom}(M) \\ \text{undefined} & \text{otherwise} \end{cases}$$

It can be shown that if M is partial computable then so is M'. Additionally, with the exception of the empty string, the complexities generated by the two machines agree. As a strict process machine is a type of process machine it follows that for all σ, $K_{M_D}(\sigma) \leq K_{M_S}(\sigma) + O(1) \leq K(\sigma) + O(1)$.

It is not immediately apparent that K_{M_D} and K_{M_S} differ in any significant way. The first result that we will prove in this paper is that the complexities K_{M_D} and K_{M_S} are different, that is they do not agree within

any additive constant. In fact we will go further and show that for any $a \in \mathbb{R}$ with $0 < a < 1$, there are infinitely many σ such that $K_{M_S}(\sigma) - K_{M_D}(\sigma) > a \log\log|\sigma|$.

In the opening paragraph, we stated that plain Kolmogorov complexity did not capture the basic concept behind Algorithmic Information Theory. This became apparent during attempts to define algorithmic randomness for real numbers.[1] The intuition is that a real α should be random if all of its initial segments are random strings (though we allow deviation by some constant amount). Thus we would like to be able to define α as random if $C(\alpha \upharpoonright n) \geq |n| - O(1)$ (where $\alpha \upharpoonright n$ is the first n bits of α).

The problem with this attempt at a definition is that Martin-Löf showed: given any d, then for *any* sufficiently long string υ there is an initial segment σ of υ such that $C(\sigma) < |\sigma| - d$ [4]. In particular, this proves that there are no reals α with the property that $C(\alpha \upharpoonright n) \geq |n| - O(1)$. Martin-Löf's proof made direct use of the fact that a Turing machine can use the bits of a description τ and additionally another $\log|\tau|$ bits from the length of τ to determine its output.

Prefix-free complexity or process complexity can be used to define randomness for a real number based on initial segment complexity [11, 2]. Levin also provided a similar characterization using monotonic complexity [7]. If we say that a real α is random if for all n, $Q(\alpha \upharpoonright n) \geq n - O(1)$ where Q is either K, or K_{M_D}, then we get a non-empty class of random reals. Now it does not matter which of the two we choose Q to be because they both give rise to *the same class of random reals*. The class is in fact the class of Martin-Löf random reals, the most commonly used notion of randomness in Algorithmic Information Theory. This shows that prefix-free machines and process machines are unable to use those extra $\log|\tau|$ bits. As the complexity K_{M_S} lies between K and K_{M_D} it follows that if we replace Q by K_{M_S}, we still get the same class of random reals. Hence strict process complexity retains the desirable property of process complexity by providing a simple characterization of the Martin-Löf random reals.

Additionally, like process complexity, strict process complexity provides a simple characterization of computable reals. A real α is computable if and only if $K_{M_S}(\alpha \upharpoonright n) \leq K_{M_S}(n) + O(1)$ where $K_{M_S}(n) = K_{M_S}(1^n)$ (1^n is the string formed by repeating 1 n times). The non-trivial direction follows because $K_{M_S}(n) \leq \log(n) + O(1)$ and $C(\alpha \upharpoonright n) \leq K_{M_S}(\alpha \upharpoonright n) + O(1)$. Hence if for some α, $K_{M_S}(\alpha \upharpoonright n) \leq K_{M_S}(n) + O(1)$, then $C(\alpha \upharpoonright n) \leq \log(n) + O(1)$ and so α is computable by a theorem of Chaitin's [4].

Martin-Löf's proof also showed that plain Kolmogorov complexity is not subadditive. That is to say, for any d there exists strings σ, τ such that $C(\sigma\tau) > C(\sigma) + C(\tau) + d$ where $\sigma\tau$ is the string formed by appending τ to the end of σ. It does this because we can take a random finite string υ that has an initial segment σ with $C(\sigma) < |\sigma| - d$. Now if τ is chosen so that $\sigma\tau = \upsilon$ then $C(\sigma\tau) \geq |\sigma| + |\tau| > C(\sigma) + d + C(\tau) - i$ where i is the length of the index of the identity function in U. As i is fixed we can make $d - i$ arbitrarily large. Thus we have that plain Kolmogorov complexity is not subadditive. On the other hand, prefix-free complexity is an example of a complexity that is subadditive.

The second result that we will prove is that both process complexity and strict process complexity are not subadditive. The proof Martin-Löf used for plain complexity cannot be adapted to process complexity. This is because given a random real α, it is true that $K_{M_D}(\alpha \upharpoonright n) \geq n - O(1)$ i.e. there are no arbitrary drops in initial segment complexity. Thus the question as to whether these complexities are subadditive requires new techniques. In particular we need to use non-random strings. The new techniques used for building and analyzing process machines introduced here may well have wider application.

[1] A real is identified as an infinite binary string.

1.1 Conventions

The set of all binary strings of length n, the set of all finite binary strings, and the set of all infinite binary strings will be denoted by $\{0,1\}^n$, $2^{<\omega}$, and 2^ω respectively. The empty string will be represented by λ. The relation $\leq$ on $2^{<\omega} \times (2^{<\omega} \cup 2^\omega)$ is defined by $\sigma \leq \tau$ if σ is an initial segment of τ. We say $\sigma < \tau$ if $\sigma \leq \tau$ and $\tau \not\leq \sigma$. If $\sigma \not\leq \tau$ and $\tau \not\leq \sigma$, then τ and σ are said to be *incomparable* written $\sigma \mid \tau$. The operation of appending a string τ to the end of a finite string σ, will be represented by $\sigma\tau$. If $\sigma \in 2^{<\omega}$ let $|\sigma|$ be the length of σ and if $i \in \mathbb{N}$ with $1 \leq i \leq |\sigma|$ let $\sigma(i)$ be the ith bit of σ.

The main proofs in this paper will be combinatorial in nature. They will make use of some basic properties of Cantor space. Cantor space is the topology on 2^ω defined by taking $\{[\sigma] : \sigma \in 2^{<\omega}\}$, where $[\sigma] = \{\sigma\alpha : \alpha \in 2^\omega\}$ for each $\sigma \in 2^{<\omega}$, as a basis of open sets. If $X \subseteq 2^{<\omega}$, then $[X] = \bigcup_{\sigma \in X}[\sigma]$. The Lebesgue measure on Cantor space μ is the outer measure obtained by defining $\mu([\sigma]) = 2^{-|\sigma|}$ for all open sets $[\sigma]$ in the basis. The main properties of Cantor space that we need are as follows. Firstly $\mu(2^\omega) = \mu([\lambda]) = 1$. Secondly if τ_1 and τ_2 are incomparable elements of $2^{<\omega}$ then $[\tau_1] \cap [\tau_2] = \emptyset$. This implies that if $A \subseteq 2^{<\omega}$ is a prefix-free set, then $\mu([A]) = \sum_{\sigma \in A} 2^{-|\sigma|}$.

Given a partial computable function $M : X \to Y$ and $x \in X$, we write $M(x)\downarrow$ if x is an element of the domain of M and $M(x)\uparrow$ otherwise. Further, if we are regarding M as a Turing machine, we will write $M(x)[s]\downarrow$ if M halts on input x within s computational steps, and $M(x)[s]\uparrow$ otherwise.

Logs used are all base 2 and are rounded up to the nearest integer value. By convention $\log 0 = 0$.

2 Strict process complexity and process complexity

In this section we will show that strict process complexity and process complexity are in fact different. As the universal strict process machine is a process machine, there is some constant d such that for all $\sigma \in 2^{<\omega}$:

$$K_{M_D}(\sigma) \leq K_{M_S}(\sigma) + d$$

We want to show that this inequality cannot be reversed and so strict process complexity and process complexity do not agree within an additive constant. To show this, we will make use of the fact that the universal strict process machine has a computable approximation. Let U be a universal strict process machine. For all $s \in \mathbb{N}$, define:

$$U_s(\tau) = \begin{cases} U(\tau) & \text{if } U(\tau)[s]\downarrow \\ \uparrow & \text{otherwise} \end{cases}$$

Because U is a strict process machine, we can take our approximation to have the property that if $U_s(\tau)\downarrow = \sigma$ and $\tau' < \tau$ then there is some $\sigma' \leq \sigma$ such that $U_s(\tau')\downarrow = \sigma'$.

In lemma 2.3, we will show that for any $i \in \mathbb{N}$, we can construct a process machine f_i such that there exists a string σ_i with $C^{f_i}(\sigma_i) + i < K_{M_S}(\sigma_i)$. Once we have done this, it will not be too difficult to combine these machines to prove that strict process complexity and process complexity do not agree within an additive constant.

To understand the ideas behind the proof of lemma 2.3, let us take the case $i = 1$ as an example. We will construct a process machine f_1. When we construct this machine, we are able to first define f_1 for all strings of length 3. Then at a later stage, we have the option of defining f_1 for strings of

length 2 and even later for strings of length 1. This option is not available to the universal strict process machine U. Once a string τ is added to the domain of U, all initial segments of τ must be added as well. Our definition of f_1 starts as follows. First let $\tau = abc$ be any binary string of length 3, i.e. a, b, and c are the first, second and third bits of τ respectively. We define f_1 by $f_1(abc) = a^8b^{16}c$ e.g. $f_1(010) = 00000000111111111111111110$. We can consider this as 'stretching' all but the last bit of the string abc.

Now we wait until some stage s_1, when for all $\tau \in \{0,1\}^3$, $C^{U_{s_1}}(f_1(\tau)) \leq 4$. If this never happens then we have finished because for some $\tau \in \{0,1\}^3$,

$$\begin{aligned} K_{M_S}(f_1(\tau)) &= \lim_{s\to\infty} C^{U_s}(f_1(\tau)) \\ &> 4 \\ &= |\tau| + 1 \\ &= C^{f_1}(f_1(\tau)) + 1 \end{aligned}$$

So assume such a stage s_1 occurs. For all $\tau \in \{0,1\}^3$, let ρ_τ be some string in the domain of U_{s_1} such that $|\rho_\tau| \leq 4$ and $U_{s_1}(\rho_\tau) = f_1(\tau)$. Let A_1 be the set of all such ρ_τ. Note that A_1 must be a prefix-free set and $|A_1| = 2^3$. If it is not, then there is some $\tau, \upsilon \in \{0,1\}^3$ with: $\tau \neq \upsilon$; $\rho_\tau, \rho_\upsilon \in A_1$; and $\rho_\tau \leq \rho_\upsilon$. But this would mean that $f_1(\tau) = U_{s_1}(\rho_\tau) \leq U_{s_1}(\rho_\upsilon) = f_1(\upsilon)$ which contradicts our definition of f_1. It follows that $\mu([A_1]) \geq |A_1|2^{-4} = \frac{1}{2}$.

We will now define f_1 for all binary strings of length 2. Given any two bit binary string ab, there must be some string a^8b^k with $1 \leq k \leq 16$ such that $C^{U_{s_1}}(a^8b^k) > 3$. This is true because there are at most 15 strings whose complexity is less than or equal to 3 (as there are only 15 such descriptions). Now we define $f_1(ab) = a^8b^k$. Now consider for a moment how the universal strict process machine can respond to this. Because U is a strict process machine, if τ is any initial segment of a string in A_1, then $U(\tau)[s_1]\downarrow$. So if $|\tau| \leq 3$, then $U(\tau) \neq f_1(ab)$ for any $a, b \in \{0,1\}$. This means that in order to reduce the complexity of $f_1(ab)$ to 3 or less, U needs to find a short description that is not an initial segment of an element of A_1.

Again we wait until some stage s_2, when for all $\tau \in \{0,1\}^2$, $C^{U_{s_2}}(f_1(\tau)) \leq 3$. If this never happens then again our objective is achieved. If this stage does occur then for all $\tau \in \{0,1\}^2$, let ρ_τ be some string in the domain of U_{s_2} such that $|\rho_\tau| \leq 3$ and $U_{s_2}(\rho_\tau) = f_1(\tau)$. Let A_2 be the set of all such ρ_τ. Again $\mu([A_2]) \geq |A_2|2^{-3} = \frac{1}{2}$. We want to show that $[A_1] \cap [A_2] = \emptyset$. Take any $\rho_1 \in A_1$ and $\rho_2 \in A_2$. $|U(\rho_1)| > |U(\rho_2)|$ so we know that $\rho_1 \not\leq \rho_2$. Now let us show that $U(\rho_2)[s_1]\uparrow$. If $U(\rho_2)[s_1]\downarrow$, then $C^{U_{s_1}}(U(\rho_2)) \leq |\rho_2| \leq 3$. So by our construction of f_1, $f_1(ab) \neq U(\rho_2)$ for any $a, b \in \{0,1\}$. This is a contradiction and so $U(\rho_2)[s_1]\uparrow$. As $U(\rho_1)[s_1]\downarrow$ so it must be that $\rho_2 \not< \rho_1$ because U is a strict process machine. Hence ρ_1 and ρ_2 are incomparable and so $\mu([\mathrm{dom}(U_{s_2})]) \geq \mu(A_1) + \mu(A_2) = 1$.

Finally we define $f_1(0) = 0^k$ for some $1 \leq k \leq 8$ such that $C^{U_{s_2}}(0^k) > 2$ (there must be some k as there are only 7 possible descriptions of length less than or equal to 2). Consider any $\upsilon \in 2^{<\omega}$. As $\mu(A_1 \cup A_2) = 1$, either υ is an initial segment, or an extension, of some element $\rho \in A_1 \cup A_2$. If $\upsilon > \rho$ and $U(\upsilon)\downarrow$ then $U(\upsilon) \geq U(\rho)$ and so $U(\upsilon) \neq f_1(0)$. If $\upsilon \leq \rho$, then $U(\upsilon)[s_2]\downarrow$ so if $U(\upsilon) = f_1(0)$ then

it must be that $|\upsilon| > 2$ (as we chose $f_1(0)$ so that $C^{U_{s_2}}(f_1(0)) > 2$). Hence:

$$\begin{aligned} C^{f_1}(f_1(0)) + 1 &= 1 + 1 \\ &< C^U(f_1(0)) \\ &= K_{M_S}(f_1(0)) \end{aligned}$$

The main idea is that if the universal strict process machine attempts to respond each time strings are added to the domain of f_1, then it will run out of measure. The key point is that during the construction of f_1, we can reuse measure by using initial segments of strings already in the domain of f_1. On the other hand, U needs to add new measure into its domain at each response. To adapt this argument to hold for any i, let us start with a lemma giving a lower bound on the measure of the domain of a strict process machine.

Lemma 2.1. *If $\{(\tau_1, \sigma_1), \ldots, (\tau_n, \sigma_n)\}$ is a set of ordered pairs such that for all $i, j \in \mathbb{N}$, $1 \leq i, j \leq n$, $U(\tau_i) = \sigma_i$; and if $i \neq j$ then:*

1. *$\sigma_i \neq \sigma_j$; and*
2. *$\sigma_i > \sigma_j$ implies that there exists an s such that $U_s(\tau_i) \downarrow$ and $U_s(\tau_j) \uparrow$;*

then $\mu([dom(U)]) \geq \sum_{i=1}^{n} 2^{-|\tau_i|}$.

Proof: We will show that the τ_i form an prefix-free set. Choose any $i \neq j$. If $\sigma_i \mid \sigma_j$ then as U is a process machine, $\tau_i \mid \tau_j$. If $\sigma_i > \sigma_j$ then there exists an s such that $U_s(\tau_i) \downarrow$ and $U_s(\tau_j) \uparrow$, so as U is a strict process machine τ_j is not an initial segment of τ_i. Further τ_j cannot extend τ_i as this would imply that $\sigma_j \geq \sigma_i$. Hence $\tau_i \mid \tau_j$. Similarly if $\sigma_j > \sigma_i$ then $\tau_i \mid \tau_j$ as well. ■

We will formalize the notion of stretching a string as follows.

Definition 2.2. If $g : \mathbb{N} \to \mathbb{N}$, then $\hat{g} : 2^{<\omega} \to 2^{<\omega}$ is defined by:

$$\hat{g}(\tau) = \tau(1)^{g(1)} \tau(2)^{g(2)} \ldots \tau(|\tau|)^{g(|\tau|)}$$

For example if g is the identity function, then $\hat{g}(0101) = 0^1 1^2 0^3 1^4 = 0110001111$.

Lemma 2.3. *For all $i \in \mathbb{N}$, there exists a process machine f_i and a string σ_i such that:*

$$C^{f_i}(\sigma_i) + i < K_{M_S}(\sigma_i).$$

Proof: Again we take U to be the universal strict process machine. We looked at the case f_1 as an example. In this case we started by defining f_1 for strings of length 3, and then for strings of progressively shorter lengths. For the general case, we will start by defining f_i for strings of length $2^i + 1$. If necessary we will define f_i for strings of length 2^i, $2^i - 1$, $2^i - 2$ and so on. Our prompt to extend the domain of f_i is if at some stage s, the universal strict process machine has made $C^{U_s}(\sigma) \leq C^{f_i}(\sigma) + i$ for all elements σ in the domain of f_i at stage s.

We define $g_i : \mathbb{N} \to \mathbb{N}$ by $g_i(n) = 2^{n+i+1}$. We will use g_i in order to stretch strings. g_i is defined like this because for any n, there are only $2^{n+i+1} - 1$ descriptions of length less than or equal to $n+i$. By making $g_i(n)$ larger than this, we know that at any stage s, we can always find initial segments of the stretched string without descriptions of length $\leq n+i$ in U_s.

The construction of f_i proceeds as follows. At stage 0, first set $l_0 = 2^i + 1$. Then for all $\tau \in \{0,1\}^{l_0}$ set $f_i(\tau) = \hat{g}_i(\tau \upharpoonright (|\tau| - 1))\tau(|\tau|)$ i.e. we use g_i to stretch all but the last bit of τ.

At stage $s+1$, if there exists some $\tau \in \{0,1\}^{l_s}$ such that $C^{U_s}(f_i(\tau)) > |\tau| + i$, then set $l_{s+1} = l_s$ and go to the next stage.

Otherwise, we know that for all $\tau \in \{0,1\}^{l_s}$, $C^{U_s}(f_i(\tau)) \leq |\tau| + i$. Now we will extend the domain of f_i. We set $l_{s+1} = l_s - 1$. For all $\tau \in \{0,1\}^{l_{s+1}}$ and for all $k \in \mathbb{N}, 1 \leq k \leq g_i(|\tau|)$. Let $\sigma_{\tau,k} = \hat{g}_i(\tau \upharpoonright (\tau - 1))\tau(|\tau|)^k$. As there are $2^{|\tau|+i+1}$ possible values of k, it follows that for any τ, there must be some k such that: $C^{U_s}(\sigma_{\tau,k}) > |\tau| + i$ because there are only $2^{|\tau|+i+1} - 1$ descriptions of length less than or equal to $|\tau| + i$. For all $\tau \in \{0,1\}^{l_{s+1}}$, let $\sigma_\tau = \sigma_{\tau,k}$ for such a k and set $f_i(\tau) = \sigma_\tau$.

To verify the construction we will first show that for all s, $l_s > 0$. We will prove this by showing that the alternative implies that U runs out of measure. If $l_s = 0$ for some s, then for all $\tau \in 2^{<\omega}$ such that $1 \leq |\tau| \leq 2^i + 1$, $K_{M_S}(f_i(\tau)) \leq |\tau| + i$ because this is the condition to extend the domain of f_i. So for all such τ we can choose a string ρ_τ such that:
(i) $|\rho_\tau| \leq |\tau| + i$,
(ii) $U(\rho_\tau) = f_i(\tau)$,
and such that no other string with properties (i) and (ii) halts before $U(\rho_\tau)$ halts.

Now take the set $A = \{(\rho_\tau, f_i(\tau)) : \tau \in 2^{<\omega}, 1 \leq |\tau| \leq 2^i + 1\}$. Consider any $\tau_1, \tau_2 \in 2^{<\omega}$, $1 \leq |\tau_1|, |\tau_2| \leq 2^i + 1$, and $\tau_1 \neq \tau_2$. First $f_i(\tau_1) \neq f_i(\tau_2)$ as f_i is injective. If $f_i(\tau_1) < f_i(\tau_2)$ then by construction this implies that $\tau_1 < \tau_2$. Now f_i is defined for τ_1 after f_i is defined for τ_2. Further if $f_i(\tau_1)$ is first defined at stage $t+1$, it must be that:

1. $C^{U_t}(f_i(\tau_2)) \leq |\tau_2| + i$; and
2. $C^{U_t}(f_i(\tau_1)) > |\tau_1| + i$.

(1) follows as this is required to extend the domain of f_i to strings of length $< |\tau_2|$. (2) follows by our choice of $f_i(\tau_1)$. Now by (i) and (ii), (1) implies that $U_t(\rho_{\tau_2}) \downarrow$ and (2) implies that $U_t(\rho_{\tau_1}) \uparrow$. The set A therefore meets the conditions of lemma 2.1 and this implies that:

$$\begin{aligned}
\mu([\mathrm{dom}(U)]) &\geq \sum_{\tau \in 2^{<\omega}, 1 \leq |\tau| \leq 2^i+1} 2^{-|\rho_\tau|} \\
&\geq \sum_{\tau \in 2^{<\omega}, 1 \leq |\tau| \leq 2^i+1} 2^{-|\tau|-i} \\
&= 2^{-i}(2^i + 1) > 1
\end{aligned}$$

A contradiction and so for all s, $l_s > 0$.

Let $n = \min\{l_s : s \in \mathbb{N}\}$. It follows that for some $\tau \in \{0,1\}^n$, for all s, $C^{U_s}(f_i(\tau)) > |\tau| + i$ and hence $K_{M_S}(f_i(\tau)) > |\tau| + i$. Thus we can take $\sigma_i = f_i(\tau)$, and we have that $C^{f_i}(\sigma_i) + i = |\tau| + i < K_{M_S}(\sigma_i)$.

Finally, f_i is a process machine because consider any $\tau_1, \tau_2 \in \mathrm{dom}(f_i)$ and $\tau_1 < \tau_2$. We have that: $f_i(\tau_1) \leq \hat{g}_i(\tau_1) \leq \hat{g}_i(\tau_2 \upharpoonright (\tau_2 - 1)) \leq f_i(\tau_2)$. ■

Note that lemma 2.3 is uniform.

Theorem 2.4. *K_{M_D} and K_{M_S} do not agree within an additive constant.*

Proof: Define a process machine f by $f(0^i 1\tau) = f_{2i}(\tau)$. Let c be the length of the index of f in the universal process machine. Now given any d, take $i = 2(c+d+1)$. By lemma 2.3, there exists some σ_i such that $C^{f_i}(\sigma_i) + i < K_{M_S}(\sigma_i)$, so:

$$\begin{aligned} K_{M_D}(\sigma_i) + d &\leq C^f(\sigma_i) + d + c \\ &\leq C^{f_i}(\sigma_i) + d + c + \frac{i}{2} + 1 \\ &= C^{f_i}(\sigma_i) + i \\ &< K_{M_S}(\sigma_i) \end{aligned}$$

■

In fact we can go further and prove that this constant can be replaced with a function of order $\log\log(n)$ where n is string length. First note that we can determine an upper bound on the length of the σ_i from lemma 2.3. Because $\sigma_i = f_i(\tau)$ for some τ with $|\tau| \leq 2^i + 1$, and as we stretch all but the last bit of τ we know that:

$$\begin{aligned} |\sigma_i| &\leq 1 + \sum_{n=1}^{2^i} g_i(n) \\ &= 1 + \sum_{n=1}^{2^i} 2^{n+i+1} \\ &= 1 + 2^{i+2} \sum_{n=0}^{2^i-1} 2^n \\ &= 1 + 2^{i+2}(2^{2^i} - 1) \\ &< 2^{2^i+i+2} \end{aligned}$$

It will be easier to express this as $\log|\sigma_i| < 2^i + i + 2$.

Theorem 2.5. *Given any $a \in \mathbb{R}$, $0 < a < 1$, then there exist infinitely many σ such that:*

$$K_{M_S}(\sigma) - K_{M_D}(\sigma) > a \log\log|\sigma|.$$

Proof: Given any such a choose $k \in \mathbb{N}^{>0}$ such that $a < 1 - \frac{1}{k}$. Now define a process $f : 2^{<\omega} \to 2^{<\omega}$ by $f(0^i 1\tau) = f_{2ki}(\tau)$. Let c be the length of the index of f in the universal process machine. Choose any $d \in \mathbb{N}$ such that $2k-1$ divides $c+d+1$. Now let $i = \frac{2k(c+d+1)}{2k-1}$. Hence i is a positive integer, and

$i = c + d + 1 + \frac{i}{2k}$. This implies that $\frac{i}{2k}$ is a positive integer too. Let $\sigma_d = \sigma_i$ from lemma 2.3. We know that:

$$\begin{aligned} K_{M_D}(\sigma_d) + d &\leq C^f(\sigma_d) + c + d \\ &= C^{f_i}(\sigma_d) + c + d + 1 + \frac{i}{2k} \\ &= C^{f_i}(\sigma_d) + i \\ &< K_{M_S}(\sigma_d) \end{aligned}$$

Also we know that:

$$\begin{aligned} \log|\sigma_d| &< 2^{\frac{2k(c+d+1)}{2k-1}} + \frac{2k(c+d+1)}{2k-1} + 2 \\ &= c_1 + \frac{2k}{2k-1}d + c_2 \cdot 2^{\frac{2k}{2k-1}d} \end{aligned}$$

where c_1 and c_2 are constant. Let $j = \frac{k}{(2k-1)(2k-2)}$. As there are infinitely many d that we can choose, we can consider those d such that $2^{jd} \geq \max(c_1 + \frac{2k}{2k-1}d, c_2, 2)$. So we have that:

$$\begin{aligned} \log|\sigma_d| &< 2^{jd} + 2^{jd}2^{\frac{2k}{2k-1}d} \\ &= 2^{jd}(1 + 2^{\frac{2k}{2k-1}d}) \\ &\leq 2^{jd}(2^{\frac{2k}{2k-1}d + jd}) \\ &= 2^{\frac{2k}{2k-1}d + 2jd} \\ &= 2^{\frac{k}{k-1}d} \end{aligned}$$

The last step follows because:

$$\begin{aligned} \frac{2k}{2k-1} + 2j &= \frac{2k}{2k-1} + \frac{2k}{(2k-1)(2k-2)} \\ &= \frac{4k^2 - 2k}{(2k-1)(2k-2)} \\ &= \frac{k}{k-1} \end{aligned}$$

So $\log\log|\sigma_d| < \frac{k}{k-1}d$ and hence we have that:

$$a \log\log|\sigma_d| < \frac{k-1}{k}\log\log|\sigma_d| < d$$

So for these such d:

$$\begin{aligned} K_{M_D}(\sigma_d) + d &< K_{M_S}(\sigma_d) \\ \Rightarrow K_{M_D}(\sigma_d) + a\log\log|\sigma_d| &< K_{M_S}(\sigma_d) \\ \Leftrightarrow K_{M_S}(\sigma_d) - K_{M_D}(\sigma_d) &> a\log\log|\sigma_d| \end{aligned}$$

There are infinitely many d that meet the conditions we require so the result follows. ■

Theorem 2.5 gives a lower bound on the difference between K_{M_D} and K_{M_S}. A basic upper bound on the difference between these complexities is that for all σ, $K_{M_S}(\sigma) - K_{M_D}(\sigma) \leq 2\log|\sigma| + O(1)$. This holds because the difference between monotone complexity K_M (a related complexity) and prefix-free complexity is bounded above by the same amount and both $K_M(\sigma) \leq K_{M_D}(\sigma) + O(1)$ and $K_{M_S}(\sigma) \leq K(\sigma) + O(1)$[10]. This leaves an open question with respect to the difference between these two complexities – which of these two bounds can be improved?

3 Process complexity is not subadditive

Our objective for this section is to prove that process complexity and strict process complexity are not subadditive. We will present the proof for process complexity but it holds without modification for strict process complexity as well.

Theorem 3.1. *Let U be a universal process machine. For any $d \in \mathbb{N}$, there exists a σ, τ such that:*

$$K_{M_D}(\sigma\tau) > K_{M_D}(\sigma) + K_{M_D}(\tau) + d.$$

We will prove this theorem by giving short descriptions to a lot of strings. We will argue combinatorially that one of these strings σ, must have an extension $\sigma\tau$ such that the desired property holds. The following lemma expresses a basic combinatorial fact about process machines.

Lemma 3.2. *If $A \subseteq 2^{<\omega}$ is a prefix-free set, then:*

$$\sum_{\sigma \in A} 2^{-K_{M_D}(\sigma)} \leq 1$$

Proof: Consider $B = \{\tau_\sigma : \sigma \in A\}$ where τ_σ is a shortest description of σ with respect to the universal process machine U. If τ_1, τ_2 are distinct elements of B, then $U(\tau_1)$ and $U(\tau_2)$ are incomparable (as they are both in A). Therefore because U is a process machine we have that $\tau_1 \not\leq \tau_2$. Hence B is an prefix-free set as well and the result follows because:

$$\sum_{\sigma \in A} 2^{-K_{M_D}(\sigma)} = \sum_{\tau \in B} 2^{-|\tau|} = \mu([B]) \leq 1$$

■

Let $g : \mathbb{N} \to \mathbb{N}$ be the constant function $g(x) = 2$. Now the function $\hat{g} : 2^{<\omega} \to 2^{<\omega}$ is a process (recall the definition of $\hat{g}$ in section 2). For all i, we can define $A_i = \{\hat{g}(\tau) : \tau \in \{0,1\}^i\}$. So $A_0 = \{\lambda\}$, $A_1 = \{00, 11\}$, $A_2 = \{0000, 0011, 1100, 1111\}$ etc.

As $\hat{g}$ is a process, there is some constant c_g such that for all $\rho \in A_i$, $K_{M_D}(\rho) \leq i + c_g = |\rho| - i + c_g$. The A_is are our sets of strings with short descriptions.

For all m, i such that $m \geq 2i+2$, we define $B_i^m = \{\sigma \in \{0,1\}^m : \exists \rho \in A_i (\rho 01 \leq \sigma \text{ or } \rho 10 \leq \sigma)\}$. The B_i^ms are the sets of extensions that we will examine. It is important to note that we have constructed the B_i^ms so that if $i \neq j$ then $B_i^m \cap B_j^m = \emptyset$. We will explain the reason for this using an example. Consider $\sigma = 00110100$. Because σ extends an element of A_2, 0011, we want to place σ in B_2^8 and not in

B_1^8 even though σ extends 00 as well. This is because if we break σ into the strings 0011 and 0100 then the difference between the length of the first string 0011, and the length of its $\hat{g}$ description is 2 ($\hat{g}(01) = 0011$). This is larger than the difference between the length of 00 and the length of its $\hat{g}$ description. We need to use the larger difference for the following argument to hold.

Note that $|B_i^m| = |A_i| \cdot 2 \cdot 2^{m-2i-2} = 2^{m-i-1}$. Now we will use the following lemma to find the extension we want.

Lemma 3.3. *Given any $e \in \mathbb{N}$, $\exists m, i, \sigma$ with $\sigma \in B_i^m$ such that $K_{M_D}(\sigma) > |\sigma| - i + e$.*

Proof: Take $m = 2^{e+3}$ and assume that no such i, σ exist. If so, then:

$$\begin{aligned}
\sum_{\sigma \in \{0,1\}^m} 2^{-K_{M_D}(\sigma)} &\geq \sum_{i=0}^{\frac{m}{2}-1} \sum_{\sigma \in B_i^m} 2^{-K_{M_D}(\sigma)} \\
&\geq \sum_{i=0}^{\frac{m}{2}-1} \sum_{\sigma \in B_i^m} 2^{-|\sigma|+i-e} \\
&= \sum_{i=0}^{\frac{m}{2}-1} |B_i^m| 2^{-m+i-e} \\
&= \sum_{i=0}^{\frac{m}{2}-1} 2^{m-i-1} 2^{-m+i-e} \\
&= (\frac{m}{2}) 2^{-e-1} \\
&> 1
\end{aligned}$$

This is a contradiction by lemma 3.2 as $\{0,1\}^m$ is a prefix-free set. ■

Proof of theorem 3.1: The identity function is a process so there is some constant c_i such that for all $\sigma \in 2^{<\omega}$, $K_{M_D}(\sigma) \leq |\sigma| + c_i$. Now given any d, choose $e = d + c_i + c_g$. From the previous lemma, there exists some m, i, υ with $\upsilon \in B_i^m$ such that $K_{M_D}(\upsilon) > |\upsilon| - i + e$. Now set $\sigma = \upsilon \upharpoonright 2i$ so $\sigma \in A_i$ and thus $K_{M_D}(\sigma) \leq |\sigma| - i + c_g$. Choose τ so that $\sigma\tau = \upsilon$. Now $K_{M_D}(\tau) \leq |\tau| + c_i$. Combining these results gives us that:

$$\begin{aligned}
K_{M_D}(\sigma) + K_{M_D}(\tau) + d &\leq |\sigma| - i + c_g + |\tau| + c_i + d \\
&= |\sigma\tau| - i + e \\
&< K_{M_D}(\sigma\tau)
\end{aligned}$$

■

Acknowledgments

The author would like to thank his PhD supervisor, Rod Downey, for helpful discussions in the development of this paper. The author would also like to thank the anonymous referees for their helpful comments. The author also acknowledges the support of the New Zealand Tertiary Education Commission.

References

[1] G.J. CHAITIN: A theory of program size formally identical to information theory. *J. ACM*, 22(3):329–340, 1975.

[2] G.J. CHAITIN: Incompleteness theorems for random reals. *Adv. Appl. Math*, 8(2):119–146, 1987. 4

[3] A.R. DAY: On the computational power of random strings. *Annals of Pure and Applied Logic*, 160(2):214–228, 2009. 3

[4] R.G. DOWNEY AND D.R. HIRSCHFELDT: Algorithmic randomness and complexity. to appear. 2, 3, 4

[5] PÉTER GÁCS: On the symmetry of algorithmic information. *Soviet Math. Dokl.*, 15, 1974.

[6] A.N. KOLMOGOROV: Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1:1–7, 1965.

[7] L.A. LEVIN: On the notion of a random sequence. *Soviet Math. Dokl.*, 14(5):1413–1416, 1973. 3, 4

[8] L.A. LEVIN: Laws of information conservation (non-growth) and aspects of the foundation of probability theory. *Problems of Information Transmission*, 10, 1974.

[9] L.A. LEVIN AND A.K. ZVONKIN: The complexity of finite objects and the development of the concepts of information and randomness of means of the theory of algorithms. *Russian Math. Surveys*, 25(6), 1970. 1, 3

[10] MING LI AND PAUL VITÁNYI: *An Introduction to Kolmogorov Complexity and its Applications*. Springer-Verlag New York, Inc., Secaucus,NJ, USA, 2nd edition, 1997. 2, 11

[11] C.P. SCHNORR: Process complexity and effective random test. *Journal of Computer and System Sciences*, 7, 1973. 1, 2, 4

[12] C.P. SCHNORR: A survey of the theory of random sequences. In *Basic Problems in Methodology and Linguistics*, pp. 193–210. D. Reidel, Dordrecht, Holland, 1977.

[13] R.J. SOLOMONOFF: A formal theory of inductive inference. *Information and Control*, 7, 1964. 3

AUTHOR

Adam R. Day
School of Mathematics, Statistics and Operations Research
Victoria University of Wellington
PO Box 600, Wellington 6140
New Zealand