Technical Report IDSIA-05-04

A Coding Theorem for Enumerable Output Machines

Jan Poland

IDSIA, Galleria 2, CH-6928 Manno-Lugano, Switzerland*

jan@idsia.ch, http://www.idsia.ch/~jan

April 29, 2004

Abstract

Recently, Schmidhuber proposed a new concept of generalized algorithmic complexity. It allows for the description of both finite and infinite sequences. The resulting distributions are true probabilities rather than semimeasures. We clarify some points for this setting, concentrating on Enumerable Output Machines. As our main result, we prove a strong coding theorem (without logarithmic correction terms), which was left as an open problem. To this purpose, we introduce a more natural definition of generalized complexity.

Keywords

Coding Theorem, Kolmogorov Complexity, Algorithmic Information Theory, Enumerable Output Machine

One of the most remarkable results in Algorithmic Information Theory is the coding theorem due to Levin ([Lev74]). It states that the shortest description of some binary string coincides with the negative logarithm of its algorithmic probability up to an additive constant. That is, roughly spoken, if there are many long descriptions for some string, then there is also a short description. Here, description is meant in the algorithmic sense: a program for some universal Turing machine that prints the string on the output tape and then halts.

*This work was supported by SNF grant 2100-67712.02.

Instead of the discrete space of strings, consider the continuous space of all (including infinite) binary sequences generated by monotone machines (not necessarily halting). This is the commonly accepted setting for studying random sequences, sequence prediction, etc. (see e.g. [LV97]). Then the coding theorem for initial parts of sequences still holds with logarithmic accuracy. Let's call this a weak coding theorem. Moreover, Gács has shown ([Gác83]) that the logarithmic correction term is necessary, i.e. a strong coding theorem cannot hold.

Recently, Schmidhuber ([Sch00, Sch02]) suggested a family of settings which is appropriate for describing finite strings *and* infinite sequences. Objects (i.e. strings or sequences) are described by *Monotone Turing Machines* (MTMs), *Enumerable Output Machines* (EOMs), or *Generalized Turing Machines* (GTMs), each machine type leading to notions of complexity and algorithmic probability. The advantage over the common approach is that now algorithmic probabilities (also called a priori probabilities) are measures instead of semimeasures (for monotone machines, this definition appears already in [ZL70]). On the other hand, the admission of infinitely running programs entails some problems, e.g. the fact that the program space must now be considered as a continuum (Example 7). Schmidhuber proved a weak coding theorem for EOMs ([Sch02, Theorem 5.3]), and he conjectured strong coding theorems without logarithmic correction term for all three machines ([Sch02, Conjecture 5.1-5.3]). It is our aim to prove the conjecture for EOMs (Theorem 9) under slightly different conditions, namely after introducing a more natural definition of complexity. We will only briefly discuss MTMs and GTMs.

We consider the space $\mathbb{B}^{\#} = \mathbb{B}^{*} \cup \mathbb{B}^{\infty}$ of all finite binary strings and infinite binary sequences. The length of some $x \in \mathbb{B}^{\#}$ is denoted by $\ell(x)$, it clearly may be infinity.

Definition 1 [Sch02] An *Enumerable Output Machine* (EOM) is a Turing machine which is allowed to edit its output tape such it increases lexicographically. Precisely, the machine has one binary input tape (read-only, one-way infinite) without blank, one or more working tapes, and one binary output tape (one-way infinite) with blank. The input tape initially contains some infinite sequence $p \in \mathbb{B}^{\infty}$, the program. The output tape is initialized with blanks. There are only three legal ways to modify the output tape:

1. Append 0 or 1 to the current bit string.
2. Switch a 0 in the current bit string to 1.
3. Switch a 0 in the current bit string to 1 and *simultaneously* reset all subsequent output tape symbols to blank.

These rules assert that the output is always a contiguous bit string, and it never decreases lexicographically.

For instance, if the output tape contains 0101, the EOM may alter it to 01011, 0111, or even to 1. If the content of the output tape x is interpreted as a binary fraction $0.x \in [0,1]$, this means that the EOM may increase this fraction. There is no restriction considering only infinite programs here, since each finite program can be completed arbitrarily. Of course, we will have to define the length of a program appropriately.

Monotone and *Generalized Machines* (MTMs and GTMs) are defined similarly [Sch02]: An MTM may only append bits to the current output string, while a GTM may edit its output tape arbitrarily. We concentrate on EOMs in the sequel.

Definition 2 [Sch02] Let T be an EOM and $p \in \mathbb{B}^\infty$ some program on its input tape. We say that the output *converges* to some $x \in \mathbb{B}^\#$, short $T(p) \rightsquigarrow x$, if for each $n \leq \ell(x)$ and $n < \infty$ there is a $t_0 \geq 1$ such that the $[T_t(p)]_{1:n} = x_{1:n}$ for all $t \geq t_0$. (Here, $x_{1:n}$ denotes the initial n bits of x, and $T_t(p)$ is the output of T for the program p after t steps of computation.)

In other words, the output converges to x if it eventually stabilizes on all prefixes of x. It is easy to see that EOMs *always* converge, since the outputs are non-decreasing and bounded by 1^∞.

Proposition 3 [Sch02] *For any EOM T and all $p \in \mathbb{B}^\infty$, there is an $x \in \mathbb{B}^\#$ such that $U(p) \rightsquigarrow x$.*

Proposition 4 [Sch02] *There is a universal EOM U. That is, for each EOM T there is $p_T \in \mathbb{B}^*$ such that for all $q \in \mathbb{B}^\infty$ we have*

$$T(q) \rightsquigarrow x \Rightarrow U(p_T q) \rightsquigarrow x.$$

See [Sch02] for a proof. For the remainder of the paper, we fix a universal EOM U.

Definition 5 Let λ be the uniform measure on $\mathbb{B}^\infty$. That is the product measure given by $\lambda(\Gamma_z) = 2^{-n}$ on a cylinder set $\Gamma_z = \{y \in \mathbb{B}^\infty : y_{1:n} = z\}$ for $z \in \mathbb{B}^n$. For $x \in \mathbb{B}^\#$, let

$$\begin{aligned} K^E(x) &= \min\{\ell(p) : \ p \in \mathbb{B}^* \ \wedge \ U(pq) \rightsquigarrow x \ \forall \ q \in \mathbb{B}^\infty\}, & (1) \\ P^E(x) &= \lambda(\{p \in \mathbb{B}^\infty : \ U(p) \rightsquigarrow x\}), & (2) \\ KP^E(x) &= -\log_2 P(x). & (3) \end{aligned}$$

Then $P^E(x)$ is the a priori probability of x and $K^E(x) \geq KP^E(x)$ is obvious. (It is also clear that the set $\{p : U(p) \rightsquigarrow x\}$ is measurable.) The complexity $K^E(x)$ is the length of the shortest program such that *all its extensions* eventually produce

x. That is, if this program is written to the input tape the rest of which may contain anything (e.g. random bits or zeros), the output of U will converge to x. Schmidhuber defines the complexity slightly differently as

$$\tilde{K}^E(x) = \min\{\ell(p) :\ p \in \mathbb{B}^* \ \wedge\ U(p) \rightsquigarrow x \ \wedge\ U(p) \text{ reads } \le \ell(p) \text{ bits}\}. \tag{4}$$

We observe immediately $\tilde{K}^E(x) \ge K^E(x)$. Note that $K^E(x)$ corresponds naturally to the shortest description of x, whereas $\tilde{K}^E(x)$ reflects an operational property. We will also use the common prefix complexity $K(x)$, which is the length of the shortest self-delimiting program that outputs x and then halts.

Remark 6 *Schmidhuber defines the algorithmic probability (2) by a sum. This is equivalent, since he interprets this sum as an integral. The following example shows that this is necessary, i.e. that a traditional sum is not sufficient to define algorithmic probability.*

Example 7 Let $q \in \mathbb{B}^\infty$ be a program and T operate on q as follows: If $q_{1:2} = 01$, then output 1 and halt. If $q_{3:5} = 001$, then output 1 and halt. Generally, if $q_{\frac{k}{2}(k-1):\frac{k}{2}(k+1)-1} = 0^{k-1}1$ for some $k \ge 2$, then output 1 and halt. If this does not happen, T runs forever and no output occurs. Let

$$A = \{q \in \mathbb{B}^\infty : T(q) \rightsquigarrow \epsilon\} = \{q : T \text{ runs forever}\}$$

(ϵ is the empty string), then

$$\lambda(A) = \prod_{k=2}^{\infty}(1-2^{-k}) = \exp\left[\sum_{k=2}^{\infty}\ln(1-2^{-k})\right].$$

From $\ln(u) \ge \frac{u-1}{u}$, we obtain $\sum_k \ln(1-2^{-k}) \ge -\sum_k \frac{1}{2^k-1}$. Since $\sum_{k\ge2}\frac{1}{2^k-1} < \ln 2$ holds, we conclude $\lambda(A) > \frac{1}{2}$. So T's contribution to the a priori probability of the empty string $P(\epsilon)$ is at least $2^{-\ell(p_T)-1}$, where p_T is the program for T. But $p_T A$ is a Cantor set in the program space, and it contains no open set. In particular, we cannot write its probability as a sum of cylinder probabilities.

Note that this example also works on GTMs. It even works on monotone machines, but only for the distribution KP^M (which is analogously defined as KP^E, see also [Sch02]), not for the universal continuous semimeasure M (see e.g. [LV97]). In fact, M *can* be written as a sum.

Other subtle difficulties are involved with the generalized complexities. For finite strings, K^E, $\tilde{K}^E$, and P^E are are limit computable (also called approximable) in the following sense: There are recursive approximations up to arbitrary accuracy, but in general the approximations are not one-sided (this would be semi-computable), nor is there a recursive estimate of the approximation error. For infinite sequences in contrast, limit computability is not obvious and possibly not even satisfied.

Proposition 8 *Let $x \in \mathbb{B}^*$, then $K^E(x)$, $\tilde{K}^E(x)$, and $P^E(x)$ are limit computable.*

Proof. Let $x \in \mathbb{B}^*$ be a finite string, then it has a simple self-delimiting description of length $\ell(x) + 2\log_2 \ell(x) + C$. Now run all programs up to this length simultaneously and approximate $\tilde{K}^E$ by the length of the shortest program that yields x in time step t without reading more than its length. If a program p outputs an extension of x, then it will be eventually discarded, so the approximations converge to $\tilde{K}^E$. The same argument applies for K^E, but we must additionally run all possible extensions of p in parallel. If a program has an extension that does not produce x, it will be eventually discarded. This shows limit computability of $K^E(x)$ and $\tilde{K}^E(x)$. We postpone the proof for $P^E(x)$, since it follows immediately from the proof of the coding theorem. □

This simple method does not work for infinite sequences, as the following example shows. Assume that we try to approximate $K^E(x)$ for some $x \in \mathbb{B}^\infty$. There are three programs satisfying $\ell(p_1) < \ell(p_2) < \ell(p_3)$. Suppose that the shortest description for x, p_2, is very slow, and $U(p_3) \rightsquigarrow x$ much faster. On the other hand, p_1 generates only some initial part, $U(p_1) \rightsquigarrow x_{1:n}$, and does not halt. Then it is not at all clear how eventually p_2 can be preferred over *both* p_1 and p_3.

Theorem 9 (Coding Theorem) *For all $x \in \mathbb{B}^\#$, we have $KP^E(x) \overset{+}{\geq} K^E(x)$. Precisely $KP^E(x) \geq K^E(x) + C$ holds with C independent of $x \in \mathbb{B}^\#$.*

Proof. If $P(x) = 0$, then there is nothing to prove, so assume $P(x) > 0$. We will code x using a binary string $s \in \mathbb{B}^*$ having $\ell(s) \overset{\pm}{=} KP^E(x)$. We can identify s with a finite binary fraction $0.s \in [0, 1)$. In order to decode s, we define an appropriate EOM T which is independent of x. Then T corresponds to a program p_T of constant length on the universal EOM U. Hence $p_T s$ is a description for x on U.

The EOM T runs all programs on U in parallel in the following way. In phase t, execute t time steps of U on all programs of length t. (In t steps, no more than t input bits can be read.) Arrange all N_t outputs $y_1^t, \ldots, y_{N_t}^t$ produced by this procedure in lexicographical order, removing duplicates. Chop off consecutive, adjacent, disjoint half open intervals $I_j^t = [l_j^t, r_j^t) \subset [0, 1)$ for each y_j, starting at the left end of $[0, 1)$ with the lexicographically smallest output y_1. Each I_j^t has length $\sum 2^{-t}$ where the sum ranges over all programs that have produced y_j^t in phase t. Thus, each output y_j^t is assigned to an interval I_j^t of length of its probability. Moreover, $\bigcup_j I_j^t = [0, 1)$, and for all j this procedure asserts $r_j^t = l_{j+1}^t$ and

$$l_j^t = 2^{-t} \cdot \left| \left\{ p : \ell(p) = 2^{-t} \ \wedge \ U_t(p) < y_j^t \right\} \right|.$$

After each phase t, we require in addition that all *previous* outputs y_j^{t-1} occur in the list $y_1^t, \ldots, y_{N_t}^t$, even if they are not generated in phase t (because the previously generated strings have been extended). In this case, the respective interval I_j^t has length zero. So for each $j \in \{1, \ldots, N_{t-1}\}$, there is a corresponding $\imath^t(j) \in \{1, \ldots, N_t\}$ with $y^t_{\imath(j)} = y^{t-1}_j$.

Claim 10 *We have $l^t_{\imath(j)} \leq l^{t-1}_j$ and $r^t_{\imath(j)} \leq r^{t-1}_j$ (recall $I^t_j = [l^t_j, r^t_j)$).*

This is easy to see: Extensions of programs generating y^{t-1}_j in phase $t-1$ either generate the same string y^{t-1}_j or some string that is lexicographically larger. Since these strings are assigned to intervals right of I^{t-1}_j, the claim follows.

Given $0.s \in [0,1)$, after phase t the output of T is y^t_j if $0.s_{1:t} \in I^t_j$. In order to evaluate this condition, the first t bits of s are sufficient. By Claim 10, we see that T may update its output after each phase t, the update never decreases lexicographically. We now distinguish the two cases $x \in \mathbb{B}^*$ and $x \in \mathbb{B}^\infty$.

Case 1: $x \in \mathbb{B}^*$. Then for sufficiently large t, there is a $j(t,x)$ such that $y^t_{j(t,x)} = x$. Set $I^t_{j(t,x)} = I^t_x = [l^t_x, r^t_x)$, then l^t_x and r^t_x decrease monotonically by Claim 10, thus converge. Denote the limits by l_x and r_x, respectively, then $r_x - l_x = P(x)$. For $\varepsilon = \frac{P(x)}{2}$, there is t_0 such that $l^t_x \leq l_x + \varepsilon$ for $t \geq t_0$. The interval $[l_x + \varepsilon, r_x)$ has length $\frac{P(x)}{2}$, thus it covers a binary interval

$$[0.s, 0.s + 2^{-\ell(s)}) \subset [l_x + \varepsilon, r_x) \subset [l^t_x, r^t_x), \tag{5}$$

where s is a string having $\ell(s) = \lceil KP^E(x) \rceil + 2$. Since $U(p_T sq) \rightsquigarrow x$ holds for any $q \in \mathbb{B}^\infty$ according to (5), the assertion follows in this case.

Case 2: $x \in \mathbb{B}^\infty$. For $n \geq 1$, let $\lambda(x_{1:n}) = \lambda(\{p \in \mathbb{B}^\infty : U(p) \rightsquigarrow x_{1:n} * \})$, i.e. the a priori probability that the output of U converges to a sequence *starting with* $x_{1:n}$. Then $\lambda(x_{1:n}) \geq P(x)$ holds for all $n \geq 1$. Let $x'_{1:n}$ be the n-bit successor of $x_{1:n}$ (e.g. $[01011]' = 01100$) and define $\tilde{l}^t_n := l^t_{x_{1:n}}$ and $\tilde{r}^t_n := l^t_{x'_{1:n}}$ (set $\tilde{r}^t_n = 1$ if $x_{1:n} = 1^n$). By Claim 10, $\tilde{l}^t_n$ and $\tilde{r}^t_n$ decrease monotonically in t. Denote the limits by $\tilde{l}_n$ and $\tilde{r}_n$, respectively, then we have $\lambda(x_{1:n}) = \tilde{r}_n - \tilde{l}_n$. Let again $\varepsilon = \frac{P(x)}{2}$ and define $\tilde{I}_n = [\tilde{l}_n + \varepsilon, \tilde{r}_n)$. Then the intervals $\tilde{I}_n$ decrease in n, and each of them has length $\lambda(x_{1:n}) - \varepsilon$. The limit

$$\tilde{I} = \bigcap_{n=1}^{\infty} \tilde{I}_n = \bigcap_{n=1}^{\infty} [\tilde{l}_n + \varepsilon, \tilde{r}_n)$$

is therefore an interval of length $\frac{P(x)}{2}$ and covers a binary interval $[0.s, 0.s + 2^{-\ell(s)}) \subset \tilde{I}$, where $\ell(s) = \lceil KP^E(x) \rceil + 2$. Then the assertion follows from

Claim 11 *For each n, there is a t_0 such that $[U_t(p_T sq)]_{1:n} = x_{1:n}$ for all $t \geq t_0$ and $q \in \mathbb{B}^\infty$.*

This is not difficult to see: For each n, there is t_0 such that $\tilde{l}^t_n \leq \tilde{l}_n + \varepsilon$ holds. For $t \geq t_0$ and any $0.sq \in [0.s, 0.s + 2^{-\ell(s)})$, we thus have $0.sq \in \tilde{I} \subset [\tilde{l}_n + \varepsilon, \tilde{r}_n) \subset [\tilde{l}^t_n, \tilde{r}^t_n)$. This establishes the claim and concludes the proof. $\square$

This proof is inspired by Schmidhuber's proof of the weak coding theorem [Sch02, Theorem 5.3]. Note that we cannot know the time t_0 from which on s points in the correct interval. Before t_0, $0.s$ may be pointing in a much smaller interval and thus require much more bits from the input tape. This is why we cannot restrict the number of bits T will read. However, as soon as $0.s$ points in the correct interval, which will eventually occur, only $\ell(s)$ bits are actually necessary, and we may even assert that only $\ell(s)$ bits are read. We thus could replace the complexity definition (1) by "the number of bits that are eventually read after throwing away all initial trials", but the formal definition of this is quite complex. On the other hand, we clearly obtain the weak coding theorem [Sch02, Theorem 5.3] for $\tilde{K}^E$ (see (4)) by additionally specifying $\ell(s)$:

Corollary 12 [Sch02] $KP^E(x) + K(KP^E(x)) \overset{+}{\geq} \tilde{K}^E(x)$.

Now we can also complete the proof of Proposition 8 and show that $P^E(x)$ is limit computable for finite $x \in \mathbb{B}^*$. Perform the parallel execution of all programs and the assignment of strings to intervals, $x \mapsto I_x^t$, as in the proof of Theorem 9. Then we have already seen $|I_x^t| \overset{t\to\infty}{\longrightarrow} P^E(x)$, which implies the assertion.

EOMs are more expressive than monotone machines. For example, there is a short program for the halting probability Ω on a universal EOM, while Ω is Martin-Löf random and therefore not efficiently describable on a monotone machine. GTMs are still more expressive than EOMs, but GTMs need not converge. This is exactly the difficulty when trying to prove an analogous coding theorem for GTMs. It is open how to overcome this problem (see also [Sch02, Conjecture 5.3]).

The difficulty in proving a strong coding theorem for MTMs is a different one. Similarly as in the case proven in [Gác83], the tree structure of $\mathbb{B}^*$ seems to allow for strings that are not efficiently codable. So we suggest that the answer to [Sch02, Conjecture 5.1] is negative – but this remains to be shown. Nevertheless, a weak coding theorem with additional logarithmic correction does hold for MTMs: $KP^M(x) + K(\ell(x)) \overset{+}{\geq} K^M(x)$ is not difficult to verify.

Acknowledgments. Thanks to Jürgen Schmidhuber and Marcus Hutter for many valuable discussions. This work was supported by SNF grant 2100-67712.02.

References

[Gác83] P. Gács. On the relation between descriptional complexity and algorithmic probability. *Theoretical Computer Science*, 22:71–93, 1983.

[Lev74] L. A. Levin. Laws of information conservation (non-growth) and aspects of the foundation of probability theory. *Problems of Information Transmission*, 10:206–210, 1974.

[LV97] M. Li and P. M. B. Vitányi. *An introduction to Kolmogorov complexity and its applications*. Springer, 2nd edition, 1997.

[Sch00] J. Schmidhuber. Algorithmic theories of everything. Report IDSIA-20-00, quant-ph/0011122, IDSIA, Manno (Lugano), Switzerland, 2000.

[Sch02] J. Schmidhuber. Hierarchies of generalized Kolmogorov complexities and nonenumerable universal measures computable in the limit. *International Journal of Foundations of Computer Science*, 13(4):587–612, 2002.

[ZL70] A. K. Zvonkin and L. A. Levin. The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. *Russian Mathematical Surveys*, 25(6):83–124, 1970.