Theoretical Computer Science 22 (1983) 71–93
North-Holland Publishing Company

ON THE RELATION BETWEEN DESCRIPTIONAL COMPLEXITY AND ALGORITHMIC PROBABILITY*

Péter GÁCS
Computer Science Department, University of Rochester, Rochester, NY 14627, U.S.A.

Communicated by A. Meyer
Received February 1981
Revised August 1981

Abstract. Several results in Algorithmic Information Theory establish upper bounds on the difference between descriptional complexity and the logarithm of 'a priori probability'. It was conjectured that these two quantities coincide to within an additive constant. Here, we disprove this conjecture and show that the known overall upper bound on the difference is exact. The proof uses a two-person memory-allocation game between players called User and Server. User sends incremental requests of memory space for certain structured items, Server allocates this space in a write-once memory. For each item, some of the allocated space is required to be in one piece, in order to give a short address. We also present some related results.

1. Introduction and the main result

In inductive inference, descriptional complexity can be used to formalize 'Occam's Razor' – the principle recommending the use of the simplest hypothesis among those consistent with the data. The principle known as 'Bayes' Rule' assumes a certain 'a priori' probability distribution over the set of possible outcomes and uses the conditional probability for inference. Algorithmic Information Theory (abbreviated henceforth as AIT) originated from the recognition that descriptional complexity, if defined appropriately, [17, 8], can be estimated by counting arguments and corresponds well to the intuitive notion of *entropy* for individual objects. (For the exact elaboration of the analogy to entropy, see [4, 3].)

Descriptional complexity was successfully used for the definition of randomness [14, 9, 11]. A priori probability, as defined in [17,-10], gives satisfactory inferences over a wide range of situations. It is a simple but central result that descriptional complexity is asymptotically equal to the negative logarithm of a priori probability [17, 10–12]. AIT owes much of its convincing power to the fact that it established this exact relation between two induction principles (Occam's Razor and Bayes' Rule) which did not seem particularly related. The main result of this paper is concerned with the exactness of this relation.

* Part of this work appeared as a Stanford Technical Report while the author visited Stanford University in 1979. An abbreviated version appeared in the proceedings of the 1981 *FOCS* conference.

Notation

Let N denote the set of natural numbers, Q the set of rational numbers, put $Z_2=\{0,1\}$. For any set A, let $A^{(n)}=\bigcup_{i=0}^{n}A^i$ be the set of strings of length $\leq n$ with elements from A. Put $A^*=\bigcup_{n\geq 0}A^n$. Let $\Lambda\in A^*$ denote the empty string and $A^\infty=A^*\cup A^N$ the set of finite or infinite strings with elements from A. We will use $\mathscr{X}=N^N$, $\mathscr{B}=Z_2^N$ for the sets of infiite strings of natural numbers and bits respectively. For $x, y\in N^*$, $x\subseteq y$ denotes that x is a prefix of y. Let $l(x)$ denote the length of the sequence $x\in N^*$ and put $x^n=x_1\cdots x_n$.

For a string $x\in N^*$, put $x\mathscr{X}=\{\omega\in\mathscr{X}: x\subseteq\omega\}$. For a set $A\subseteq N^*$, put $A\mathscr{X}=\bigcup_{x\in A}x\mathscr{X}$. The operations $p\mathscr{B}$, $A\mathscr{B}$ are defined analogously. For a binary sequence $p\in\mathscr{B}$, let $[p]\in[0,1]$ denote the real number associated with it in the binary number system. For $A\subseteq Z_2^*$, put $[A]=\{[\omega]:\omega\in A\mathscr{B}\}$. For $p\in Z'^*_2$, put $[p]=[\{p\}]$. For $E, F\subseteq N^*$, put $E'=\{x\in E:\forall y\in E\, y\subseteq x\Rightarrow y=x\}$ and $E\leq F\Leftrightarrow\forall x\in E\ \exists y\in F\, y\subseteq x$. The relation $E\leq F$ implies $E\mathscr{X}\subseteq F\mathscr{X}$ but the converse is not true.

Let $\langle\cdot\rangle: N^*\mapsto N$ be some standard one-to-one encoding with partial inverses pr_i defined by $\mathrm{pr}_i(\langle x\rangle)=x_i$ for $l(x)\geq i$. *For* $l(x)=2$, we use $\langle x\rangle$ as a two-argument pairing function $\langle x_1, x_2\rangle$.

The relations $f\leq g$, $f\asymp g$ denote $f\leq g+O(1)$, $f-g=O(1)$. All logarithms and exponentials in this paper are to the base 2. Let $\#A$ denote the number of elements of the set A.

'Algorithmic entropy' or 'self-information' seems a more appropriate term than 'descriptional complexity' partly because there are many different notions of 'complexity'. But since different quantities can claim to measure the true algorithmic entropy, we abide by the term (descriptional) 'complexity'. Even for descriptional complexity, several different definitions are intuitively almost equally justified (and generally asymptotically equal); but some bring more sharpness into the basic formulas and simplicity into the proofs than others. Most of our definitions and basic facts are taken from [10, 11]. We first tell how to *interpret* a description.

Definition 1.1. A function $A: Z_2^*\times N^*\mapsto N^*\cup N^N$ monotonic in both arguments w.r. to $\subseteq$ is a *monotonic operator* (interpreter) if $\{(p, x, y): y\in N^*, y\subseteq A(p, x)\}$ is recursively enumerable.

Put $A(p)=A(p,\Lambda)$. The first element $(A(p))_1$ of the string $A(p)$ is a natural number, which will be denoted by $A^1(p)$. The monotonicity condition for $A^1(p)$ reduces to the following: $A^1(p)=n, p\subseteq q\Rightarrow A^1(q)=n$. This seems weaker than the 'self-delimiting' requirement in [12, 3] but leads to the same complexity. We will write $A_t(p, x)$ for the part of $A(p, x)$ occurring in t steps of enumeration.

Definition 1.2. (see [11, 12]). For $x, y\in N^\infty$,

$$K_A(y\,|\,x)=\min\{l(p): y\subseteq A(p, x)\}$$

is the (monotonic) *conditional complexity* of y with respect to x and the interpreter A.

It is known that there is an *optimal* interpreter U with respect to which complexity is minimal to within an additive constant. Thus, for any interpreter A, there is a constant c_A such that for all x, y, $K_U(y\,|\,x) \leqslant K_A(y\,|\,x) + c_A$. Let us fix an optimal interpreter U, write $K(y\,|\,x) = K_U(y\,|\,x)$. The number $K(x) = K(x\,|\,\Lambda)$ is simply called the *complexity* of x. The first complexity taking into account the partial order of the finite sequences was proposed in [13] (it is different from K).

Typical orders of magnitude: for natural numbers n (sequences of length 1), $K(n) \leqslant 2\log n$, moreover, $K(n) \leqslant \log n + K(\lfloor \log n \rfloor)$. This last estimate is sharp for most numbers $k \leqslant n$ (see e.g. [3]).

For a monotonic operator A and $x \in N^*$, put $D_A(x) = \{p \in Z_2^* : x \subseteq A(p)\}$. The work of a 'Turing machine' with a read-only tape moving in, working tapes, and a write-only tape moving out where all tapes are capable of holding arbitrary natural numbers in their cells, can be represented by a monotonic operator with the special property that the set $\{(p, x) : p \in (D_A(x))'\}$ is recursively enumerable. The complexity K' based on such machines might conceivably be a little larger than K. However, all known upper bounds apply as well to K'.

The notion of a random sequence was introduced in [14]. In [11], monotonic complexity is used to characterize randomness. (These results are refined in [5].) But randomness is more immediately characterizable using *a priori probability*. Let $\pi = \pi_1, \pi_2, \ldots$ be an infinite sequence of identically distributed random variables with $\Pr[\pi_1 = 0] = \Pr[\pi_1 = 1] = \frac{1}{2}$.

Definition 1.3. The *a priori probability* of the string x is

$$M(x) = \Pr[x \subseteq U(\pi)].$$

The number $M(x)$ is the probability that our optimal machine produces x from a random input. The following representation brings out the relation to complexity.

$$\begin{aligned} M(x) &= \sum \{2^{-l(p)} : p \in (D(x))'\}, \\ 2^{-K(x)} &= \max\{2^{-l(p)} : p \in (D(x))'\}. \end{aligned} \tag{1.1}$$

The a priori probability is not a probability measure of $\mathscr{X}$ because on some strings $p \in \mathscr{B}$, the sequence $U(p)$ does not have infinite length. A nonnegative real function ν over N^* is a *semimeasure* if $\nu(\Lambda) \leqslant 1$ and for all $x \in N^*$,

$$\sum_{n \geqslant 0} \nu(xn) \leqslant \nu(x).$$

A semimeasure ν is a *measure* if equality holds here, a *probability* measure if also $\nu(\Lambda) = 1$. Put $\mu(x\mathscr{X}) = \mu(x)$. Then μ can be uniquely extended to the σ-algebra

generated by sets of the form $x\mathscr{X}$ (the Borel sets). In this way we arrive at a measure as defined in standard measure theory (see e.g. [7]). The Lebesgue-measure λ over Z_2^* is defined by $\lambda(p)=2^{-l(p)}$. When ν is restricted to natural numbers (sequences of length 1) then the semimeasure property simplifies to $\sum_n \nu(n)\leq 1$.

Put $\nu^{(n)}(x)=\sum\{\nu(y): x\subseteq y\in N^n\}$, $\bar{\nu}(x)=\lim_{n\to\infty}\nu^{(n)}(x)$. The function $\bar{\nu}$ is a measure which is maximal among all measures $\mu\leq\nu$. The statement that a set S has *a priori probability* 0 means $\bar{M}(S)=0$. The statement that a property holds for *a priori almost all* ω means that the set of all ω for which it does not hold has a priori probability 0. For any semimeasure ν and set $S\subseteq N^*$ put $\nu(S)=\sum_{x\in S'}\nu(x)$. Then for any measure μ we have $\mu(S)=\mu(S\mathscr{X})$. The following properties, expressing some restricted monotonicity and additivity properties for semimeasures, are useful:

$$S_0\leq S_1 \Rightarrow \nu(S_0)\leq\nu(S_1), \qquad \nu\Big(\bigcup_n S_n\Big)\leq\sum_n \nu(S_n). \tag{1.2}$$

The semimeasure $M(x)$ is not computable but has some weaker computability property. A real function f is called *semicomputable* (from below) if there exists a recursive function $g: N^*\times N\mapsto Q$ nondecreasing in its second argument such that $f(x)=\lim_{t\to\infty} g(x,t)$ (g *generates* f). The a priori probability M is semicomputable because $M(x)=\lim_{t\to\infty} M_t(x)$ where $M_t(x)=\Pr[x\subseteq U_t(\pi)]$. (Notice that M_t itself is a semimeasure.) A function f is *computable* if f and $-f$ are semicomputable. Semicomputable [semi]measures will also be called r.e. (recursively enumerable). It is known that r.e. *probability measures* are also computable.

All r.e. semimeasures can be *effectively enumerated* in a sequence ψ_e ($e\in N$). Indeed, let $T: N^3\mapsto Q$ be a universal partial recursive function. It is known that there exists a recursive function $g(e)$ with the following properties:

(1) For each fixed e, the function $T(g(e),x,t)$ generates a r.e. semimeasure ψ_e;

(2) If $T(e,x,t)$ generates a r.e. semi-measure then $T(g(e),x,t)=T(e,x,t)$.

It is known that $M(x)$ majorizes all r.e. semimeasures to within a multiplicative constant:

$$M(e)\psi_e(x)\leq M(x)O(1). \tag{1.3}$$

For the generation of *computable* semimeasures, we nead a *pair* of sequences: approximations from below and above. We denote by η_e the computable semimeasure (if it exists) that is generated by the pair $T(\mathrm{pr}_1(e),x,t)$, $T(\mathrm{pr}_2(e),x,t)$.

Put $H(x)=-\log M(x)$. It follows from (1.1) that $H(x)\leq K(x)$. Below, we use Martin-Löf's notion of randomness.

Fact 1.1. (see [11]).

(*a*) *For any computable semimeasure* η_e,

$$K(x)\leq -\log\eta_e(x)+K(e). \tag{1.4}$$

(*b*) *A sequence* $\omega\in\mathscr{X}$ *is random with respect to a computable measure* η_e *iff there exists a constant* c_ω *such that for all* n, $-\log\eta_e(\omega^n)\leq H(\omega^n)+c_\omega$.

Hence for sequences ω random with respect to some computable measure (certainly a wide class) it is known that $K(\omega^n)-H(\omega^n)$ is bounded by some additive constant depending on ω. Levin raised the conjecture in [11] that $H(x)\succcurlyeq K(x)$ for all $x\in N^*$. In the main result of this paper, we refute this conjecture.

A set $E\subseteq N^*$ is called *prefixfree* if its elements are not prefixes of each other. (Example: the set N^n of sequences of length n.) It is known (see [12]) that for any r.e. prefixfree set E a constant c_E exists such that we have $|H(x)-K(x)|\leqslant c_E$ for all $x\in E$. It is enough to prove this for $E=N$, i.e. that

$$H(n)\succcurlyeq K(n) \tag{1.5}$$

for natural numbers n; the rest follows by encoding. Hence

$$H(x)\leqslant K(x)\leqslant H(x)+K(l(x)) \tag{1.6}$$

since only $K(l(x))$ additional bits are needed to define the prefixfree set $N^{l(x)}$. The estimate

$$H(x)\leqslant K(x)\leqslant H(x)+K(\lfloor H(x)\rfloor) \tag{1.7}$$

– which is somewhat better for binary sequences – is proved analogously. These results show that only the tree-structure of N^* can cause a significant difference between $H(x)$ and $K(x)$. (Of course, the problem is equivalent for Z_2^*.) We will prove

Theorem 1.1. *For any function $g: N\mapsto N$ semicomputable from above for which*

$$K(x)-H(x)\leqslant g(l(x)) \tag{1.8}$$

we have $K(n)\leqslant g(n)$.

Notice that for functions $g(n)$ semicomputable from above, $K\leqslant g$ is equivalent to $\sum_n 2^{-g(n)}<\infty$. Indeed, it follows from $H(x)\leqslant K(x)$ that $\sum_n 2^{-K(n)}\leqslant\sum_n M(n)\leqslant M(\Lambda)\leqslant 1$. Hence $K\leqslant g$ implies $\sum_n 2^{-g(n)}<\infty$. On the other hand, if $\sum_n 2^{-g(n)}<\infty$ then for some constants c and e, $c2^{-g(n)}=\psi_e(n)$. Hence by (1.5) and (1.3), we have $K(n)\leqslant H(n)\leqslant g(n)$.

The strings x giving the lower bound may contain very large numbers. Therefore for binary strings, the lower bound obtainable from the proof of Theorem 1.1 is only the inverse of some version of Ackermann's function.

Theorem 1.1 shows that in the worst case, the difference between K and H can be large. On the other hand, Theorem 2.3 shows that for a priori almost every ω, $K(\omega^n)-H(\omega^n)$ has an upper bound which is smaller than any unbounded recursive function.

2. Information in largeness

2.1. Complexity of large numbers

The power of a notation for numbers can be measured by the size of the largest number describable by strings of given length. Let

$$\alpha(n) = \min_{n \leq i} K(i) = \min\{l(p): n \leq U^1(p)\}$$

be the length of the shortest description of a number larger than n. Then $\lim_{n\to\infty} \alpha(n) = \infty$ since $\#\{n: K(n) < k\} < 2^k$. The function $\alpha(n)$ grows slower than any recursive function, since there is no nonconstant recursive lower bound on $K(n)$ ('Berry's paradox'). Its inverse is a version of the 'busy beaver' function (see [15]).

The following informal remarks are intended to show that these functions play an illuminating role in AIT. The formal exposition is continued in the next subsection. For any set $E \subseteq N$, let the natural number $E(n)$ be the standard encoding of the first 2^n elements of the sequence which is the characteristic function of E. It is proved in [1] that if E is r.e. then $K(E(n)|n) \leq n$ and for a suitable r.e. set F, we have $K(F(n)|n) \asymp n$.

The information stored in $F(n)$ is algorithmically equivalent to a description of *large numbers*. Indeed: let $\{x(1), x(2), \ldots\}$ be a recursive enumeration of the set F, put $F_t = \{x(1), \ldots, x(t)\}$ and $\rho(n) = \min\{t: F_t(n) = F(n)\}$. Then, given n, knowing $F(n)$ is the same as knowing any number larger than $\rho(n)$. It is easy to see that

$$\alpha(\rho(n)|n) \asymp n,$$

i.e. $\rho(n)$, the size of the numbers "described" in $F(n)$, has 'approximately' the same order of growth as the inverse of $\alpha(n)$. (We get the definition of $\alpha(n|x)$ by conditionalizing the definition of $\alpha(n)$.)

The number $F(n)$, whose binary encoding has length 2^n, is not the shortest description of the large numbers it encodes since it contains only n bits of information. Minimal definitions of large numbers are algorithmically equivalent to prefixes of the binary expansion Ω of the number $\sum_{n\in N} M(n)$. Indeed, it is shown in [3] that $K(\Omega^n) \asymp n$ therefore Fact 1.1(b) implies that Ω is random in the Martin-Löf sense. Let $\sigma(n)$ be the time needed to approximate $[\Omega]$ within 2^{-n}. Given n, knowing Ω^n is the same as knowing a number larger than $\sigma(n)$, and

$$\alpha(\sigma(n)|n) \leq n \leq \alpha(\sigma(n)).$$

The string Ω^n can thus be considered a compressed form of $F(n)$.

The redundancy in $F(n)$ is not useless. The number $F(n)$ contains, in easily accessible form, all significant information about the *results* of computations performable in time $\alpha^{-1}(n)$. It is not surprising therefore that the computational complexity of any significant compression of $F(n)$ is nonrecursively large (see [1]). For a popular exposition of this topic, see [2, 6].

2.2. Probability of large numbers

We will consider another natural 'busy beaver' function associated with the a priori probabiltiy. For any semimeasure ν, put

$$s(n;\nu) = -\log\Big(\sum_{i\geq n} \nu(i)\Big),$$

$s(n) = s(n;M)$. Then $2^{-s(n)} = \Pr[n \subseteq U(\pi)]$ is the probability of obtaining a number larger than n using a random input to U. We have $2^{-s(0)} = [\Omega]$. Put $C(n) = \bigcup_{k\geq n} D(k)$. We have

$$2^{-s(n)} = \lambda(C(n)) = \sum\{2^{-l(p)} : p \in (C(n))'\}.$$

$$2^{-\alpha(n)} = \max\{2^{-l(p)} : p \in (C(n))'\}.$$

A comparison with (1.1) shows that the relation of $\alpha(n)$ to $s(n)$ is analogous to the relation of $K(x)$ to $H(x)$. In analogy to (1.7), it is shown in [18] that

$$s(n) \leq \alpha(n) \leq s(n) + K(\lfloor s(n) \rfloor). \tag{2.1}$$

We will show that the error term $K(\lfloor s(n) \rfloor)$ is necessary in (2.1).

Theorem 2.1. *Let $g: N \mapsto N$ be a monotonic function semicomputable from above such that*

$$\alpha(n) \leq s(n) + g(\lfloor s(n) \rfloor). \tag{2.2}$$

Then $K(n) \leq g(n)$.

Note. Since we required monotonicity, $\log n + K(\lfloor \log n \rfloor) \leq g(n)$ is also proved as $\log n + K(\lfloor \log n \rfloor)$ is $\asymp$ to the least monotonic upper bound on $K(n)$. (This follows from Theoerem 4.2(a) of [3].)

The proof of Theorem 2.1 is given in Section 3.

2.3. Properties of a priori almost all sequences

How fast can a sequence increase if it is generated by a probabilistic Turing machine? The next two results of L.A. Levin and N.V. Petri have not been published before in this form.

Theorem 2.2. *For a priori almost all ω there is a constant C such that $\alpha(\omega_n) \leq 2\alpha(n) + K(\alpha(n)) + C$.*

Having a bound on the growth of random sequences, we also have an esimate of the closeness of M_t to M since the minimal times t giving $M_t = M$ are also 'random'. Combining this remark with (1.4), we get an estimate of $K - H$.

Theorem 2.3. *For a priori almost all ω, there is a constant C such that*

$$K(\omega^n) - H(\omega^n) \leq 2\alpha(n) + K(\alpha(n)) + C.$$

In the rest of this section, I prove Theorems 2.2 and 2.3.

Lemma 2.1. *For any semimeasure ν and measure $\mu \leq \nu$ put $S_m = \{x \in N^*: 2^m \mu(x) < \nu(x)\}$. Then $\mu(S_m) \leq 2^{-m}$. If $\nu(\Lambda) - \mu(\Lambda) < \varepsilon$ then $\nu(S_2) < 2\varepsilon$.*

Proof. Put $\delta(x) = \nu(x) - \mu(x)$. δ is a semimeasure with $\delta(\Lambda) < \varepsilon$. The condition $x \in S_m$ translates into $\mu(x) \leq 2^{-m}\nu(x)$, while $x \in S_2$ also translates into $\nu(x) < 2\delta(x)$. Summing over S'_m gives $\mu(S_m) \leq 2^{-m}\nu(S_m)$ and $\nu(S_2) < 2\delta(S_2)$. Using $S_m \leq \{\Lambda\}$ and (1.2) concludes the proof. □

Lemma 2.2. *Let ν be a semimeasure. For a sequence $S(m) \in N^*$ of sets and $r \in N$, put $S = \bigcap_m S(m)\mathscr{X}$ and*

$$S(m, r) = \{x \in N^r: \exists n\, \alpha(n) = K(r), x^n \in S(m)\}.$$

Then $\nu(S(m, r)) = O(1)M(r)2^{-m}$ implies $\bar{\nu}(S) = 0$.

Proof. Put $S_1(m) = \bigcup_r S(m, r)$. Notice that $S(m)\mathscr{X} = S_1(m)\mathscr{X}$. Using (1.2), we have $\bar{\nu}(S(m)) \leq \nu(S_1(m)) \leq \sum_r \nu(S(m, r)) = O(1)2^{-m} \sum_r M(r) = O(1)2^{-m}$. □

Proof of Theorems 2.2 and 2.3. For some $m, r \in N$, put $k = K(r)$. Let Ω^r_{km} be the sequence of the first $k + m$ digits in the binary expansion of $M(N^r)$. Put

$$t = t(m, r) = \min\{u: [\Omega^r_{km}] < M_u(N^r)\},$$

$$E_0(m, r) = \{x \in N^r: \exists y \subseteq x\ 2M_t^{(r)}(y) < M^{(r)}(y)\}.$$

for $x \in N^{(r)}$, the function $M_t^{(r)}(x)$ behaves like a measure. By the definition of t, the function $M^{(r)}(\Lambda) - M_t^{(r)}(\Lambda) < 2^{-k-m}$. Therefore Lemma 2.1 can be applied and we get

$$M(E_0(m, r)) = M^{(r)}(E_0(m, r)) < 2^{-k-m+1}. \tag{2.3}$$

The complexity of $t(m, r)$ can be estimated as follows. We have $K(t) \leq K(r) + K(\langle \Omega^r_{km} \rangle)$. To give Ω^r_{km}, we need a binary string of length $k + m$ along with a description of the number $k + m$. This takes $K(k + m) \leq K(k) + K(m) \leq K(k) + m$ bits. Hence

$$K(t) \leq 2k + K(k) + 2m. \tag{2.4}$$

The inequalities (2.3) and (2.4) are the key relations for the derivations of both theorems. Let us start with Theorem 2.2. Put

$$b(k) = 2k + K(k),$$

$$s = s(m, r) = \max\{x_n: x \in N^*, n \leq r, M_t(x) > 0\}.$$

Since s is defined using t, $K(s) \leqslant K(t)$. Hence it follows from (2.4) that there is a C_1 such that $K(s) < b(k) + 2m + C_1$. Put

$$E_1(m, r) = \{x \in N^r : \exists n\, \alpha(n) = k,\ \alpha(x_n) > b(k) + 2m + C_1\}.$$

If $\alpha(x_n) > b(k) + 2m + C_1$, then, by (2.4), we have $\alpha(x_n) > K(s)$, hence $x_n > s$. Hence $0 = 2M_t^{(r)}(x) < M^{(r)}(x)$. So we proved $E_1(m, r) \subseteq E_0(m, r)$, hence $M(E_1(m, r)) < 2^{-k-m+1} = \mathrm{O}(1) M(r) 2^{-m}$. Put

$$E_1(m) = \{x \in N^* : \alpha(x_{l(x)}) > b(\alpha(l(x))) + 2m + C_1,\ \},$$

$$E_1 = \bigcap_m E_1(m) \mathscr{X} = \{\omega \in \mathscr{X} : \forall m\ \exists n\, \alpha(\omega_n) > b(\alpha(n)) + m\}.$$

Lemma 2.2 is applicable to the sets $E_1(m)$ and the conclusion, $\bar{M}(E_1) = 0$, is the assertion of Theorem 2.2.

Now we prove Theorem 2.3. It follows from (2.4) that the shortest description p of t has length $\leqslant b(k) + 2m$. The semimeasure $M_t = M_{U(p)}$ is computable, moreover, by an application of the S_m^n-theorem of recursion theory, it can be written as $\eta_{f(p)}$ for some recursive function f. Therefore, by (1.4), using $K(f(p)) \leqslant K(p)$ we get $K(x) \leqslant -\log M_t(x) + K(p) \leqslant -\log M_t(x) + b(k) + 2m$. Hence for some constant C_2,

$$K(x) < -\log M_t(x) + b(k) + 2m + C_2 \tag{2.5}$$

holds for all $x \in N^{(r)}$. Let $F_1(m, r)$ be the set of those $x \in N^r$ for which there is an n with $\alpha(n) = k$ and $K(x^n) > -\log \bar{M}(x^n) + b(k) + 2m + C_2 + 1$. If $x \in F_1(m, r)$ then for some $y \subseteq x$, using $M_t^{(r)} \leqslant M_t$, inequality (2.5) and $\bar{M} \leqslant M^{(r)}$, we have $-\log M_t^{(r)}(y) > -\log M_t(y) > -\log \bar{M}(y) + 1 \geqslant -\log M^{(r)}(y) + 1$. Hence $F_1(m, r) \subseteq E_0(m, r)$ and $M(F_1(m, r)) = \mathrm{O}(1) M(r) 2^{-m}$. Put

$$F_1(m) = \{x \in N^* : K(x) > -\log \bar{M}(x) + b(\alpha(l(x))) + 2m + C_2\}.$$

As in Lemma 2.2, we have $\bar{M}(F_1(m)) = \mathrm{O}(1) 2^{-m}$. Put

$$F_0(m) = \{x \in N^* : 2^m \bar{M}(x) < M(x)\}.$$

By Lemma 2.1, $\bar{M}(F_0(m)) \leqslant 2^{-m}$. Put

$$F_2(m) = \{x \in N^* : K(x) > H(x) + b(\alpha(l(x))) + 3m + C_2\}.$$

Then $F_2(m) \subseteq F_0(m) \cup F_1(m)$. Hence $\bar{M}(F_2(m)) = \mathrm{O}(1) 2^{-m}$. Put

$$F_2 = \bigcap_m F_2(m) \mathscr{X} = \{\omega : \forall m\ \exists n\, K(\omega_n) > H(\omega_n) + b(\alpha(l(x))) + m\}.$$

We proved $\bar{M}(F_2) = 0$. □

3. Weights on large numbers

This section is devoted to the proof of Theorem 2.1. We introduce an infinite 2-person game. In this section, we consider only semimeasures over N, therefore a *semimeasure* is simply a nonnegative function $a(x)$ over N with $\sum_{x\geq 0} a(x)\leq 1$.

Game 1

This game has no memory-allocation interpretation but for greater uniformity, we use a terminology similar to the one used in Sections 4 and 5. The game is determined by the nonnegative functions $a_t(x)$ and $b_t(x)$ defined on $N\times N$ and nondecreasing in t. Let $a_0(x)=b_0(x)=0$ for all x. For fixed t, both functions a_t and b_t are semimeasures and have rational values different from 0 only for finitely many x. The number $b_t(x)$ is a power of $\frac{1}{2}$. At step $t+1$, the players User and Server know a_t and b_t. User chooses a_{t+1} and Server replies with b_{t+1}. Put $a=\lim_{t\to\infty} a_t$ and $b=\lim_{t\to\infty} b_t$. Server wins if for all n, $\max_{k\geq n} b(k)$ is large enough relative to $\sum_{k\geq n} a(k)$. Precisely, the result of the game depends on a function $g:N\mapsto N$. Put

$$\alpha(n;b)=\dot{-}\log\max_{k\geq n} b(k).$$

Then Server wins if for all n,

$$\alpha(n;b)\leq s(n;a)+g(\lfloor s(n;a)\rfloor). \tag{3.1}$$

Theorem 3.1. *Let $g:N\mapsto N$ be a function semicomputable from above.*
(a) *Server has a recursive strategy such that if*

$$\sum_{n\geq 0} 2^{-g(n)}\geq\tfrac{1}{4} \tag{3.2}$$

then he wins.
(b) *if g is monotonic and*

$$\sum_{n\geq 0} 2^{-g(n)}=\infty \tag{3.3}$$

then User has a recursive strategy making (3.1) *fail, moreover, which for some i, n gives*

$$s(n;a)+g(i)\leq i<\alpha(n;b). \tag{3.4}$$

Remark. The inequality (3.4) implies that (3.1) fails, because then $\lfloor s(n;a)\rfloor\leq i$. By the monotonicity of g, hence $s(n;a)+g(\lfloor s(b;a)\rfloor)<\alpha(n;b)$.

Let us apply this theorem to the proof of (2.1). Take $g(n)=K(n)+2$. Then g is semicomputable from above and (3.2) holds. Put $a_t(x)=M_t(x)$. Then $a_t(x)$ is a recursive function of t and x. Let Server apply his winning strategy. Then $b_t(x)$ is

also recursive and both $M(x)=a(x)$ and $b(x)$ are r.e. semimeasures. We have therefore

$$\alpha(n)\leqslant\alpha(n;b)\leqslant s(n;a)+g(\lfloor s(n;a)\rfloor)=s(n)+K(\lfloor s(n)\rfloor)+2.$$

Now we prove Theorem 2.1 using Theorem 3.1. Let $g(n)$ be a monotonic function semicomputable from above for which (3.3) holds. We show that (2.2) does not hold. For every natural number k, put $g^k(n)=g(n)+k$. Obviously, if g satisfies (3.3) then so does g^k. Let User play the recursive strategy a_t^k which makes (3.1) fail for some n when g^k is used. Put $b_t(x)=\exp\lfloor\log M_t(x)\rfloor$. Since $b_t(x)$ is a recursive function and $a_t^k(x)$ depends only on it, $a_t^k(x)$ is also a recursive function. Hence a^k is a r.e. semimeasure which, by the S_m^n-theorem, can be written as $\psi_{f(k)}$ for some recursive function f. Using (1.3) and $M(k)=O(M(f(k)))$ gives $a^k(x)=O(M(x)/M(k))$. With some constant C_1, we have for some i and n:

$$\alpha(n)+1\geqslant\alpha(n;b)>i\geqslant s(n;a^k)+g^k(i)\geqslant s(n)+g(i)+k-H(k)-C_1.$$

For k sufficiently large, we have $k-H(k)-C_1>0$, hence $\lfloor s(n)\rfloor\leqslant i$, so $g(\lfloor s(n)\rfloor)\leqslant g(i)$, $\alpha(n)>s(n)+g(\lfloor s(n)\rfloor)$ which contradicts (2.2).

Proof of Theorem 3.1. Since g is semicomputable from above, there is a recursive function $g_t(n)$ of t, n nonincreasing in t with $g(n)=\lim_{t\to\infty}g_t(n)$.

Proof of (*a*): Suppose that (3.2) holds. The strategy of Server is to handle every possible value of $\lfloor s(n;a_t)\rfloor$ separately. Multipliers of $2^{-g_t(i)}$ are needed to maintain $\sum_x b_t(x)\leqslant 1$. Suppose a_t, b_{t-1} are given. Put

$$i_t=\min\{i:\exists n\ \lfloor s(n;a_t)\rfloor=i,\ \alpha(n;b_{t-1})>i+g(i)\}.$$

Let $n(t)$ be the minimal n for which the above minimum is achieved (if at all. Else put $i_t=\infty$.) Put $j_b(t)=1+\max\{j:a_t(j)+b_{t-1}(j)>0\}$,

$$b_t(x)=\begin{cases}2^{-i_t-g_t(i_t)} & \text{for } x=j_b(t),\\ b_{t-1}(x) & \text{otherwise.}\end{cases}$$

By the definition of this strategy of Server, we will have

$$\alpha(n;b_t)\leqslant\lfloor s(n;a_t)\rfloor+g_t(\lfloor s(n;a_t)\rfloor),$$

hence in the limit, (3.1) for all n. We have to prove $\sum_x b(x)\leqslant 1$. If $b(x)>0$ then for some t we have $x=j_b(t)$ and $b(x)=b_t(x)=2^{-i_t-g(i_t)}$. Put $T_{ir}=\{t:i=i_t,\ g_t(i)=r\}=\{t_1,\ldots,t_{k_{ir}}\}$. We have

$$\sum_x b(x)=\sum_i\sum_{r\geqslant g(i)}k_{ir}2^{-i-r}.$$

If we can show that $k_{ir}\leqslant 2^{i+1}$ then we are done because then summing over r gives $\leqslant 2^{-g(i)+2}$, and summing over i and using (3.2) gives $\leqslant 1$. Notice that for all p, $j_b(t_p)<n(t_{p+1})$. Indeed, by definition, we will have $\alpha(n;b_t)\leqslant i+r$ for all $n\leqslant j_b(t_p)$, $t>t_p$. Therefore, since $\lfloor s(n(t_{p+1});a_{t_{p+1}})\rfloor=i$, the sum of a must increase by at least

2^{-i-1} between t_p and t_{p+1}:

$$2^{-i-1} \leqslant \sum_x a_{t_{p+1}}(x) - \sum_x a_{t_p}(x).$$

The above inequality holds also for $p=0$ if we put $t_0=0$. Since the sum of a_t can never increase above 1, summing over p gives $k_{ir} \leqslant 2^{i+1}$.

Proof of (*b*): Suppose that (3.3) holds. The game terminates if $\sum_x a_t(x)=1$. The strategy of User uses two constans u and m to be chosen at the end of the proof. At each step t, User does nothing if for some i, n,

$$s(n; a_t) \leqslant i - g_u(i), \tag{3.5}$$

$$i < \alpha(n; b_t). \tag{3.6}$$

Otherwise, he will place a new weight of 2^{-m} after the tails of a_t and b_t as long as he can. Formally, put $j_a(t) = 1 + \max\{j: a_t(j) + b_t(j) > 0\}$,

$$a_{t+1}(x) = \begin{cases} 2^{-m} & \text{for } x = j_a(t), \\ a_t(x) & \text{otherwise.} \end{cases}$$

With this strategy, if the game never terminates then, (3.4) holds for some i, n. Indeed, for some t_0, $a_t = a_{t_0}$ for all $t \geqslant t_0$. Put $i(n) = \min\{i:$ (3.5) holds for $t = t_0\}$. Then $S = \{n: i(n) < \infty\}$ is finite. If for infinitely many t there is some i, n for which (3.5)–(3.6) holds then for some $n \in S$, (3.6) holds with $i = i(n)$ and all t. So (3.4) follows.

Suppose therefore that the game halts at $t = T$. Then for each t, there is a $t' \geqslant t$ such that a does not change between t and t' and for all n, i, $s(n; a_t) \leqslant i - g_u(i)$ implies $\alpha(n; b_{t'}) \leqslant i$. We can assume w.l.o.g. that $t' = t$, i.e.

$$s(n; a_t) \leqslant i - g_u(i) \Rightarrow \alpha(n; b_t) \leqslant i. \tag{3.7}$$

We will show that to guarantee (3.7), Server must handle the different values i of $\lfloor s(n; a_t) \rfloor$ separately, just as in the proof of (a). Put

$$E_i = \{t: \exists n\ 2^{-i} \leqslant b_t(n),\ b_{t-1}(n) < b_t(n)\}.$$

At each time $t \in E_i$, Server uses at least 2^{-i-1} of his total weight of 1 to increase some $b(n)$. Our goal is to show that the weight used up this way is of the order of $2^{-g_u(i)}$. Let us fix i and write $E_i = \{t_1, \ldots, t_{s_i}\}$ where $t_p < t_{p+1}$. Put $t_0 = 0$, $t_{s_i+1} = T+1$. We will estimate s_i from below by estimating the weight $\sigma_p = 2^{-m}(t_{p+1} - t_p - 1)$ assigned by User at stages between t_p and t_{p+1}. While $t_p < t < t_{p+1}$, the value $b_t(n)$ cannot increase from 0 to $\geqslant 2^{-i}$ for any n. Hence $i < \alpha(j_a(t_p); b_{t_{p+1}-1})$ and from (3.7), if $\sigma_p \neq 0$ then

$$i - g_u(i) \leqslant s(j_a(t_p); a_{t_{p+1}-1}) = -\log \sigma_p,$$

hence $\sigma_p \leqslant 2^{-i+g_u(i)}$. However, the total weight assigned by User is 1. Hence

$$1 = s_i 2^{-m} + \sum_p \sigma_p \leqslant (s_i + 1) 3^{-i+g_u(i)+1}$$

holds for $i \geq m$. Hence $s_i \geq 2^{i-g_u(i)-1}-1$ for $i \leq m$. Since $\sum_x b_t(x)$ will increase by at least 2^{-i-1} for $t \in E_i - E_{i-1}$, we have

$$1 \geq s_0 2^{-1} + \sum_{i=1}^{m} (s_i - s_{i-1}) 2^{-i-1} = \sum_{i=1}^{m} s_i 2^{-i-1} - \sum_{i=1}^{m-1} s_i 2^{-i-2}$$

$$\geq \sum_{i=1}^{m-1} s_i 2^{-i-2} \geq \sum_{i=1}^{m-1} (2^{-g_u(i)-3} - 2^{-i-2})$$

$$\geq \frac{1}{8} \sum_{i=1}^{m-1} 2^{-g_u(i)} - 1.$$

It is clear now from (3.3) that User can choose u and m large enough to get a contradiction. □

4. A simple storage-allocation game

We describe a storage-allocation game between two players called User and Server to investigate (1.4) and see why does an analogous result not hold over N^*.

Game 2

User sends at each step $t \in N$ some quantities of some items. The set of items is identified with N. Item 0 can be vacuum cleaners, item 1 towels, item 3 oil, etc. Let the real number $a(x, t) \geq 0$ be the total quantity sent until time t from item x. Let $a(x, 0) = 0$. At step t, the function $a(x, t)$ has rational values different from 0 only for finitely many $x \in N$. We have $\sum_x a(x, t) \leq 1$, i.e. $a(x, t)$ is a semimeasure (see Section 1) for all t over N. The function $a(x, t)$ is monotonically increasing in t.

Server has a store, the interval $[0, 1]$, where at each step t, every point $[\omega] \in [0, 1]$, is allocatable to some item $x = A(\omega, t)$. The partial function $A(\omega, t)$ has the property that for each x and t, the set $B(x, t) = \{\omega : x = A(\omega, t)\}$ (the storage space allocated to x) is a finite union of intervals, nonempty only for finitely many x. We extend the definiton of $A(\cdot, t)$ to $Z_2^* \cup Z_2^N$ by $A(p, t) = x \Leftrightarrow [p] \subseteq B(x, t)$. Then for each t, the function $A(\cdot, t)$ is a monotonic operator. Server is never allowed to reallocate (e.g. the store is a write-once memory), i.e. $A(\cdot, t+1)$ is an extension of $A(\cdot, t)$. Put $b(x, t) = \lambda(B(x, t))$ where λ is the Lebesgue measure over $[0, 1]$. It is natural to require

$$b(x, t) \geq a(x, t) \tag{4.1}$$

To satisfy (4.1), Server can begin allocating at the left end of the store and add new intervals to any $B(x, t)$ as $a(x, t)$ is increasing. We suppose that Server must allocate a large part of $B(x, t)$ in one piece. If the store is interpreted as memory, then this gives x a location with a short address. In AIT, this address can be used as a short description of x.

Remark. I do not claim any immediate applicability of the games described here in memory-allocation strategies of contemporary operating systems. However, the condition that memory is not reallocatable is sometimes fulfilled, e.g. in allocating catalogue numbers to subject areas in libraries. It is natural to require that a large subject area receive a short number (address).

Put

$$c(x,t)=2^{-K_{A(\cdot,t)}(x)}=\max\{\lambda([p]):[p]\subseteq B(x,t)\}.$$

Let $a(x)$, $B(x)$, $b(x)$ and $c(x)$ be the respective limits of $a(x,t)$, $B(x,t)$, $b(x,t)$ and $c(x,t)$ when $t\to\infty$. Part (a) of Theorem 4.1 states that $a(x,t)/c(x,t)$ can be bounded by a constant if $w(t)=\sum_x a(x,t)$ is kept away from 1 at a constant distance. Even if no bound on $w(t)$ is available, the quotient is $O(\log a(x,t))$. Part (b) asserts that the estimate in (a) is sharp.

Theorem 4.1. (a) *Server has a recursive strategy guaranteeing*

$$\frac{a(x,t)}{c(x,t)}\leqslant -11\log\max\{a(x,t),1-w(t)\}+O(1)$$

for all x, t.

(b) *There is a constant C_1 such that for any $\varepsilon\in(0,\frac{1}{2})$, User has a recursive strategy forcing*

$$\max\left\{\frac{a(x,t)}{c(x,t)}:\varepsilon<a(x,t),1-w(t)\right\}\geqslant C_1\log\frac{1}{\varepsilon}.$$

Let us prove (1.5) using (a). Put $a(x,t)=M_t(x)/2$ for any natural number x. Let Server use his recursive strategy to produce $A(p,t)$ for $p\in Z_2^*$. Then $A(p,t)$ will also be recursive. Then $A(p)=\lim_t A(p,t)$ is a monotonic operator,with values $x\in N$. As $w(t)\leqslant\frac{1}{2}$ for all t,

$$K(x)\leqslant K_A(x)=-\log c(x)\leqslant -\log a(x)+O(1)=H(x)+O(1).$$

Proof of (a).

Lemma 4.1. *Server has a strategy to achieve $a(x,t)/c(x,t)\leqslant 2$ while $w(t)\leqslant\frac{1}{2}$. The strategy is: for all x and m, allocate to x the first empty binary interval of length 2^{-m} as soon as $a(x,t)$ passes 2^{-m} – else do nothing.*

For a simple proof that this algorithm works (i.e. that the desired binary interval will always be found) see e.g. Theorem 3.2 in [3]. The strategy in Theorem 4.1(a) is based on the following ideas.

(1) A new contiguous binary interval $I(x,t)$ is *reserved* for item x at times t when $a(x,t)$ passes an integer power of $\frac{1}{2}$. Storage is allocated for x in the reserved interval as long as it lasts. *Reservation is weaker than allocation: it can be cancelled.*

Put $s_i = 1-2^{-i}$ and $t_i = \min\{t: s_{i-1} \leqslant w(t)\}$. We can suppose that $w(t_i) = s_{i-1}$, since Server can split his answers into parts. Put $i(t) = \max\{i: t \geqslant t_i\}$. Until time t_i, only the interval $[0, s_i)$ is used.

(2) The strategy is restarted at each t_i (reservations are cancelled) and from that time on, only the *increments* in $a(x, t)$ will count.

Put $u(x, t) = \lceil -\log(a(x, t) - a(x, t_{i(t)}))\rceil$. Notice that if $u(x, t+1) < u(x, t)$ then the increase in $a(x, t)$ since t_i has passed some power of $\frac{1}{2}$. Suppose that we are after step $t+1$ of User and Server has to answer. Put $i = i(t)$. We will have $\bigcup_x B(x, t) \subseteq [0, s_{i+1})$, $I(x, t) \subseteq [s_i, s_{i+1})$, $I(x, t) \cap I(x', t') \neq \emptyset \Rightarrow x = x'$, $t = t'$ and $\bigcup_y B(y, t) \cap I(x, t) \subseteq B(x, t)$. We can suppose that $a(x, t+1) = a(x, t)$ for all but one item, x_t (Server can always divide one step into many when answering). The algorithm:

1. (Allocate as much storage for x_t in $I(x, t)$ as you can.) Put $\delta = a(x_t, t+1) - a(x_t, t)$. Let α be the measure of the free area in $I(x_t, t)$. Allocate storage of the amount $\delta_0 = \min\{\delta, \alpha\}$ to x_t in $I(x_t, t)$.
2. (Reserve a new interval for x_t if needed.) Set $I(x, t+1) = I(x, t)$ for all $x \neq x_t$. If $u(x_t, t+1) < u(x_t, t)$, set $I(x_t, t+1)$ to be the first empty binary interval of length $2^{-u(x_t, t+1)-2}$ in the nonreserved area of $[s_i, s_{i+1})$. Else, $I(x_t, t+1) = I(x_t, t)$.
3. Set $\delta_1 = \min\{\delta - \delta_0, \lambda(I(x_t, t+1))\}$. Allocate storage of the amount δ_1 for x in $I(x_t, t+1)$.
4. Allocate for the remaining quantity $\delta - \delta_0 - \delta_1$ of item x_t a union of intervals of this total length in the unused area of $[0, s_i)$. Remember that $[0, s_i)$ is large enough to hold everything until t_{i+1}.

This algorithm works if the new binary interval of the desired length can always be found for $I(x, t+1)$. But on $[s_i, s_{i+1})$, we used the strategy of Lemma 4.1, which is known to work. Now we must bound $a(x, t)/c(x, t)$. Put $\gamma_i(x) = a(x, t_i) - a(x, t_{i-1})$ for $1 \leqslant i \leqslant i(t)$ and $\gamma_{i(t)+1} = a(x, t) - a(x, t_{i(t)})$. Then

$$a(x, t) = a(x, t) - a(x, t_{i(t)}) + \sum_{j=1}^{i(t)} (a(x, t_j) - a(x, t_{j-1})) = \sum_{j=1}^{i(t)+1} \gamma_j.$$

Using our method of allocation, a binary interval of length 2^{-u-2} will be reserved in time interval (t_j, t_{j+1}) to x when $a(x, t) - a(x, t_j)$ passes 2^{-u} and will be filled up by the time this quantity passes $2^{-u} + 2^{-u-2} = \frac{5}{4}2^{-u}$. Until this time a binary interval of length 2^{-u-3} was completely allocated to x. Hence $\frac{1}{10}\max_j \gamma_j \leqslant c(x, t)$. We also know $\gamma_j \leqslant 2^{-j}$. Minimization of the function $\max_j \gamma_j$ of $(\gamma_1, \ldots, \gamma_{i(t)+1})$ subject to $\gamma_j \leqslant 2^{-j}$ gives the desired lower bound on $c(x, t)$ in terms of $a(x, t) = \sum_j \gamma_j$ and $i(t) = \lceil -\log(1 - w(t))\rceil$. □

We do not prove (b) before reformulating it in Lemma 5.1(a). A related result, Lemma 5.1(b) is needed for the proof of Theorem 1.1.

5. Storage allocation for a tree

Game 3 is an elaboration of Game 2 in which the set of items has a hierarchical structure.

Game 3

User sends requests for storage of different sorts of items. These sorts form a *hierarchical structure*: we identify them with N^*. For example, item 3 means 'cars', item 30 'Fords', 311 'subcompact AMC cars', etc. Here, the amount $a(x, t)$ of storage requests sent until time t for item x is a *semimeasure* over N^*, different from 0 at only finitely many places. The function $a(x, t)$ is nondecreasing in t. Server has a store $V \subseteq [0, 1]$ which is a finite union of intervals.

The allocation by Server at time t is described by the monotonic operator $A(p, t)$ defined for $[p] \subseteq V$. All points of the binary interval $[p]$ are allocated to x at time t iff $x \subseteq A(p, t)$. The set $B(x, t) = \{[\omega]: x \subseteq A(\omega, t)\}$ allocated to item x is a finite union of intervals, nonempty for only finitely many x. The partial function $A(\cdot, t+1)$ is an extension of $A(\cdot, t)$. The quantities $b(x, t)$, $a(x)$, $b(x)$, $c(x, t)$, $c(x)$ are defined as in Game 2.

As in Game 2, if only $a(x) \leq b(x)$ is required and $V = [0, 1]$, Server can allocate everything. This strategy translates into the theorem in [10] asserting that for every r.e. semimeasure ν there is a monotonic operator A such that $\nu(x) = \Pr[x \subseteq A(\pi)]$, i.e. that every r.e. semimeasure is the output distribution of some Turing machine with the coin-tossing distribution as input. However, Server must satisfy $\log(a(x)/c(x)) \leq g(l(x))$ for some function g. We can suppose that he must satisfy

$$\log \frac{a(x, t)}{c(x, t)} \leq g(l(x)) \tag{5.1}$$

since otherwise User could wait until (5.1) does not hold. Moreover, the relation $a(x) \leq b(x)$ *is not required.*

Theorem 5.1. *Suppose that* $V = [0, 1]$.

(a) *Server has a recursive strategy to achieve*

$$\log \frac{a(x)}{c(x)} \leq \min\{K(l(x)), K(\lfloor -\log a(x) \rfloor)\}.$$

(b) *For any function* g *semicomputable from above with* $\sum_n 2^{-g(n)} = \infty$, *User has a recursive strategy achieving* $g(l(x)) < \log(a(x)/c(x))$ *for some* x.

Theorem 5.1 implies (1.6), (1.7) and Theorem 1.1 in the way Theorem 3.1 implies Theorem 2.1 and Theorem 4.1 implies (1.5). One would be interested in the size of the smallest x with $k < \log(a(x)/c(x))$ as a function of k in (b). From the proof,

we get $l(x)=O(2^k)$. But the sequence x may contain very large numbers. Therefore we can bound $1/a(x)$ for such an x only by a version of Ackermann's function.

Proof. The positive part (a) of this theorem can be proved easily. The set N^n of all sequences of a fixed length n is prefixfree, hence on it, the simple strategy of Lemma 4.1 applies. Server sets aside a store of size $O(M(n))$ for N^n, which is feasible even though $M(n)$ can be computed only gradually. The set $G(n)=\{x: \lfloor -\log a(x)\rfloor = n\}$ is though not necessarily prefixfree but is 'almost' so – therefore almost the same procedure is applicable to get the bound $K(\lfloor -\log a(x)\rfloor)$.

The proof of the negative part (b) uses the technique of the proof of the negative part of Theorem 4.1. This result says that for some constant C, even in the prefixfree case, to get (4.1) and (5.1) we need a surplus store of the size $\exp(-C2^k)$. (It will turn out that at several points during the game, a surplus store of size $O(2^{-k})$ is needed.) This extra space is *not allocated* during the game. But it will be too *fragmented* to use for the needed large contiguous intervals. In Game 3, property (4.1) is not required but due to the hierarchical structure of items, a fragmented area may look like an allocated area for items on a higher level.

Let us formalize the idea of reservation. For any natural numbers $r\leqslant s$ and a set $E\subseteq V$ put

$$L_r^s(E;V)=\bigcup\{[p]\subseteq V: r\leqslant l(p)\leqslant s, [p]\cap E\neq\emptyset\}.$$

Put $L_r^s(E)=L_r^s(E;[0,1])$. The set $L_r^s(E;V)$ is the union of all binary intervals in V with lengths between 2^{-s} and 2^{-r} having nonempty intersection with E (made unusable by E for certain kinds of reservation). Put

$$W_r^s(t;V)=L_r^s\Big(\bigcup_x B(x,t);V\Big),$$

$$B_r^s(x,t;V)=L_r^s(B(x,t);V-\bigcup\{B(y,t): y\not\subset x \text{ and } x\not\subset y\}),$$

$w_r^s(t;V)=\lambda(W_r^s(t;V))$ and $b_r^s(x,t;V)=\lambda(B_r^s(x,t;V))$. Notice that $w(t)$ is controlled by User but $W_r^s(t;B)$ – which is also monotonic in t – is controlled by Server. The set $B_r^s(x,t;V)$ is the union of binary intervals of certain lengths which we can consider as reserved for x at time t. It is not monotonic in t: reservations may be 'cancelled'. However, reservation in one stage, even if cancelled, may look as irrevocable allocation from the point of view of subsequent stages. To implement this idea we must prepare Game 2 in a form applicable as a recursion step in Game 3.

Game $2\frac{1}{2}$

The set of items is the set N^2 of sequences of length 2. The store is $V\subseteq[0,1]$. The following additional parameters are given: an infinite nonincreasing sequence $r_0\geqslant r_1\geqslant\cdots$ of natural numbers, r, $k\in N$ with $k\leqslant r\leqslant r_i$, and a real number $\gamma>0$. Put $\Gamma=\max\{1,\gamma\}$, $\sigma=\Gamma^{-1}2^{-r-2}$. The rules are those of Game 3 and additionally

the following. We have $a(x, t) = \sum_y a(xy, t)$ for all $x \in N$ and

$$a(x, t) \leqslant 2^{-r+k}. \tag{5.2}$$

User is permitted to change $a(xy, t)$ only in a special fashion. He chooses a pair $x_t y_t$ and puts

$$a(xy, t+1) = \begin{cases} a(xy, t) + \delta(t) & \text{for } xy = x_t y_t, \\ a(xy, t) & \text{otherwise.} \end{cases}$$

The number $\delta(t) \in [\sigma, 2\sigma]$ is not chosen by User: we can suppose it is chosen by Server. Server must satisfy

$$a(x, t) = 2^{-r+k} \Rightarrow c(x, t) \geqslant 2^{-r} \tag{5.3}$$

for all $x \in N$, and also the following weak version of (4.1): for $x, y \in N$, if $a(xy, t) = 0$ and $a(xy, t+1) > 0$ then

$$b^{r_t}_{r_{t+1}}(xy, t+1) \geqslant \gamma a(xy, t+1). \tag{5.4}$$

The game ends at some time T. Requirements (5.2) and (5.3) mean that the size of contiguous intervals requested in this game is always 2^{-r}. Put

$$s_k = 3 + \Gamma 2^{k+3}.$$

Lemma 5.1. *In Game $2\frac{1}{2}$, User has a strategy with the following properties. As long as Server is able to keep the rules,*

(a)

$$w^{r_0}_r(t; V) \geqslant \gamma w(t)(1 + 2^{-s_k}) \tag{5.5}$$

for all t with $w(t) \geqslant 2^{s_k - r}$;

(b) *for any $v \in [2^{k-r+2}, w(T)/2]$ there is a t' with $w(t') \in [v, 2v]$ and*

$$w^{r_0}_r(t'; V) \geqslant w(t')(\gamma + 2^{-k-3}).$$

Proof of Theorem 4.1(b). Put $\gamma = 1$ and $r = \lfloor -\log \varepsilon \rfloor + k$. When the set of items is restricted to N, the rules of Game $2\frac{1}{2}$ are harder for User and easier for Server than the rules of Game 2. By Lemma 5.1(a), Server can keep the rules of Game $2\frac{1}{2}$ until $1 - \varepsilon < w(t)$ only if $(1-\varepsilon)(1+2^{-s_k}) < 1$, i.e. $2^{-s_k} < \varepsilon/(1-\varepsilon) < 2\varepsilon$. If $2^{-s_k} = 2\varepsilon$ i.e. $2^k = 2^{-3}(-\log \varepsilon - 4)$ then he can still keep the rules of Game 2 but there will be an x, t with $\varepsilon < a(x, t) = 2^{-r+k}$ and $a(x, t)/c(x, t) > 2^k = 2^{-3}(-\log \varepsilon - 4)$, which is the assertion of Theorem 4.1(b). □

Proof of Lemma 5.1. We give the strategy of User. Put $a(xy, 0) = 0$ for all $x, y \in N$. To determine $a(x, t+1)$, User orders all items $x \in N$ in decreasing order of values of $a(x, t)$ (for equal values of $a(x, t)$, the smaller x comes first). Let $R(x, t)$ be the rank of x in this order. For his decision, User looks up the first item x_t in this order which has no binary interval of length 2^{-r} 'reserved' for it, i.e. for which $B^r_r(x, t; V) =$

$\emptyset$. From now on, we suppress the argument V: it serves as a parameter. Put $y_t = \min\{y : B(x_t y, t) = \emptyset\}$.

We must prove (5.5). Put $F(t) = \bigcup_x B_r^r(x, t)$, $D(u, t) = F(u) \cap F(t)$, $d(t) = \lambda(F(t))$ and $C(t) = \bigcup_{u=1}^{t} B_{r_u}^{r_u - 1}(x_u y_u, u)$. First, we prove that for all t,

$$w_r^{r_0}(t) \geq \gamma w(t) + d(t)/2. \tag{5.6}$$

Obviously $W_r^{r_0}(t) \supseteq C(t) \cup F(t)$. We will prove by induction on $u \leq t$ that

$$\lambda(C(u) \cup D(u, t)) \geq \gamma w(t) + \lambda(D(u, t))/2 \tag{5.7}$$

which clearly implies (5.6). The inequality (5.7) is true for $u = 0$. Suppose that it holds for u. Suppose first that $D(u+1, t) = D(u, t)$. Then by the choice of $x_u y_u$,

$$B_{r_{u+1}}^{r_u}(x_u y_u, u+1) \cap (C(u) \cup D(u, t)) = \emptyset.$$

Therefore the left side of (5.7) increases by at least $b_{r_{u+1}}^{r_u}(x_u y_u, u+1) \geq \gamma\delta(u)$ while the right side increases exactly by $\gamma\delta(u)$, Hence (5.7) holds for $u+1$. Suppose now that $E = D(u+1, t) - D(u, t) \neq \emptyset$. Then $E \cap C(u) = \emptyset$, therefore the left side of (5.7) increases by at least $\lambda(E)$ which is a positive integer multiple of 2^{-r}. Using $\gamma\delta(u) \leq 2\sigma\gamma \leq 2^{-r-1}$, we have

$$\begin{aligned} \lambda(E) &= \lambda(E) - \gamma\delta(u) + \gamma\delta(u) \\ &\geq (\lambda(D(u+1, t)) - \lambda(D(u, t)))/2 + \gamma\delta(u) \end{aligned}$$

hence (5.7) holds for $u+1$. This proves (5.7) and hence (5.6). The inequality (5.6) gives

$$R(x_t, t) \leq 2^r d(t) \leq 2^{r+1}(w_r^{r_0}(t) - \gamma w(t)). \tag{5.8}$$

Inequality (5.8) says that if the difference between space reserved (therefore in some sense spoiled) and the space 'actually allocated' (reserved on some lower level) is small then $a(x, t)$ can increase only for some x of low rank.

Proof of (*a*): Suppose that for some t_0 with $w(t_0) \geq 2^{s_k - r}$, $\theta > 0$ we have

$$w_r^{r_0}(t_0) \leq (1+\theta)\gamma w(t_0).$$

We will show that then $2^{-s_k} \leq \theta$. Put

$$t_i = \min\{t : w(t) \geq w(t_0) - (2^i - 1)2^{-r-1}\}.$$

The number t_i is defined for all $i \leq \lfloor \log w(t_0) \rfloor + r + 1 = i_1$ and is decreasing in i. For all $t \leq t_0$, put

$$m(t) = \lfloor \gamma 2^{r+1}((1+\theta)w(t_0) - w(t)) \rfloor.$$

By (5.8), the number $m(t)$ is an upper bound on $R(x_t, t)$. Notice that $m(t)$ is a decreasing function of t. Since only the weight of items x with $R(x, t) \leq m(t)$ can increase at time t, the set $\{x : R(x, t) \leq m(t)\}$ is a decreasing function of t. Therefore the average

$$p(t) = \frac{1}{m(t)+1} \sum \{a(x, t) : R(x, t) \leq m(t)\}$$

increases. Between t_i and t_{i-1}, the rank $R(x_t, t)$ can take at most $m(t_i)+1$ different values. Using $1 \leqslant \Gamma$ we have

$$m(t_i)+1 \leqslant \Gamma + \Gamma 2^{r+1}((1+\theta)w(t_0) - w(t_i)) \leqslant \Gamma(2^i + \theta w(t_0) 2^{r+1}). \tag{5.9}$$

The weight distributed over these x_t in this time interval is $w(t_{i-1}) - w(t_i)$. We have

$$\begin{aligned} &w(t_{i-1}) \geqslant w(t_0) - 2^{-r-1}(2^{i-1}-1), \\ &w(t_i) \leqslant w(t_0) - 2^{-r-1}(2^i - 1) + 2^{-r-1}, \\ &w(t_{i-1}) - w(t_i) \geqslant 2^{-r-1}(2^{i-1}-1) \geqslant 2^{i-r-3} \end{aligned} \tag{5.10}$$

for $i > 1$. Therefore for $1 < i \leqslant i_1$, combining (5.9) with (5.10) gives

$$\begin{aligned} p(t_{i-1}) - p(t_i) &\geqslant \frac{1}{m(t_i)+1}(w(t_{i-1}) - w(t_i)) \geqslant \frac{2^{-r-3}\Gamma^{-1}}{1+\theta w(t_0) 2^{r+1-i}} \\ &\geqslant \Gamma^{-1} 2^{-r-3}(1 - \theta w(t_0) 2^{r+1-i}). \end{aligned}$$

Put $i_0 = \max\{2, \lceil \log \theta w(t_0) 2^{r+1} \rceil\}$. By (5.2), we have

$$2^{k-r} \geqslant p(t_0) \geqslant \Gamma^{-1} 2^{-r-3} \sum_{i=i_0}^{i_1} (1 - \theta w(t_0) 2^{r+1-i}).$$

If $i_0 = \lceil \log \theta w(t_0) 2^{r+1} \rceil$ then the above sum is $> \log \theta^{-1} - 3$. Rearrangement gives $2^{-s_k} < \theta$. If $i_0 = 2$ then the above sum is $> \log w(t_0) + r - 3$. Rearrangement gives $w(t_0) < 2^{s_k - r}$.

Proof of (*b*): Put

$$m(t_0, t_1) = \max\{2^r d(t) : t \in [t_0, t_1)\}.$$

By the first inequality in (5.8), the set $E(t_0, t_1) = \{x : R(x, t) \leqslant m(t_0, t_1)\}$ is independent of t and $x_t y_t \in E(t_0, t_1)$ for all $t \in [t_0, t_1)$. Therefore

$$w(t_1) - w(t_0) \leqslant 2^{k-r} \# E(t_0, t_1) \leqslant (1 + m(t_0, t_1)) 2^{k-r}.$$

Suppose that $v \in [2^{k-r+2}, w(T)/2]$. Put $t_0 = \min\{t : w(t) \geqslant v\}$, $t_1 = \min\{t : w(t) \geqslant 2v\}$ and $m = m(t_0, t_1)$. Then

$$v - 2^{-r-1} \leqslant w(t_1) - w(t_0) \leqslant (1+m) 2^{k-r}.$$

The maximum m is achieved for some $t' \in [t_0, t_1)$. We have $w(t') \in [v, 2v)$ and

$$\begin{aligned} d(t') = m 2^{-r} &\geqslant 2^{-k}(v - 2^{-r-1}) - 2^{-r} \\ &\geqslant 2^{-k-2}(4v - 2^{-r+1} - 2^{k-r+2}) \geqslant 2^{-k-2} w(t'). \end{aligned}$$

With (5.6), this completes the proof. □

Lemma 5.1 states that in the 2-level Game $2\frac{1}{2}$, User can force Server to devote to reservation significantly more area than the amount $\gamma w(t)$ required by the rules. (At least $\gamma(1+2^{-s_k})$ – always, $\gamma + s^{-k-3}$ – sometimes.) A recursive application of

the strategy of Lemma 5.1(b) will give this effect repeatedly. Suppose that the set $V \subseteq [0, 1]$ is all the store space available. Put

$$\gamma(i, g) = \sum_{j=0}^{i} 2^{-g(2j+1)-3}.$$

The recursive functions below are, strictly speaking, functionals: they depend on a function argument g.

Lemma 5.2. *There is a recursive function $f(i, r, g)$ such that User has a strategy $S(i, r, g, V)$ in Game 3 with the following properties: for all i, $g: N \mapsto N$, $r \geq g(1)+3$, if Server satisfies (5.1) for all x with $l(x) \leq 2i+1$ then for all $v \in [2^{g(1)-r+2}, \frac{1}{2}]$ there is a t' with $w(t') \in [v, 2v]$,*

$$w_r^{f(i,r,g)}(t'; V) \geq \gamma(i, g) w(t'). \tag{5.11}$$

Proof of Theorem 5.1(b). Let $g = \lim_t g_t$ be a function semicomputable from above, with $\sum_n 2^{-g(n)} = \infty$. We can suppose w.l.o.g. that $\sum_{j \geq 0} 2^{-g(2j+1)} = \infty$. We apply the strategy $S(i, g_u(1)+3, g_u, [0, 1])$. If Server satisfies (5.1) then with $v = \frac{1}{4}$ we get

$$1 \geq w_r^{f(i,r,g_u)}(t'; [0, 1]) \geq \gamma(i, g_u)/4$$

which is a contradiction for i, u sufficiently large. □

Proof of Lemma 5.2. The proof goes by induction on i. For $p \in N$, certain times t_p, will have special significance. Let us call the course of strategy $S(i, r, g, V)$ between t_p and t_{p+1} the *pth macrostep* of this strategy. For $i = 0$, User will raise $a(p, t_p)$ from 0 to 2^{-r-1} in step $t_p = p$. The inequality (5.1) guarantees (5.11). Suppose that User has a strategy $S(i, r, g, V)$ satisfying the requirements of the lemma. We have to define $S(i+1, r, g, V)$. Put $\bar{g}(n) = g(n+2)$, $\gamma = \gamma(i, \bar{g})$, $\Gamma = \max\{1, \gamma\}$, $k = g(1)$, $\sigma = \Gamma^{-1} 2^{-r-2}$ and $T = \lceil \sigma^{-1} \rceil$. We define the function f by the following double recursion:

$$f(0, r, g) = r,$$

$$f(i+1, r, g) = h(T, i, r, g),$$

where

$$h(0, i, r, g) = \lceil -\log \sigma \rceil + g(3) + 2,$$

$$h(j+1, i, r, g) = f(i, h(j, i, r, g), g).$$

Put $r_j = h(\max\{T-j, 0\}, i, r, g)$. The 'sum' of moves of User in the pth macrostep in $S(i, r, g, V)$ is on N^2 the same as his move in step p of the winning strategy in Game $2\frac{1}{2}$. Namely, the weight of some $x_p y_p$ increases by some $\delta(p) \in [\sigma, 2\sigma]$. The pth macrostep itself is an application of strategy $S(i, r_{p+1}, \bar{g}, V_p)$ to the tree of

continuations of $x_p y_p$, where

$$V_p = V - \bigcup_{u=0}^{p-1} B(x_u y_u, t_u; V)$$

is the remaining store after p macrosteps. By the definition of h, we have $r_p = f(i, r_{p+1}, g)$ for all p with $w(t_{p+1}) \leq 1$. In $S(i, r_p, \bar{g}, V_p)$, the game is played until a point t' is reached with $\bar{w}(t') \in [\sigma, 2\sigma]$,

$$\bar{w}^{r_p}_{r_{p+1}}(t'; V_p) \geq \gamma(i, \bar{g})\bar{w}(t'), \tag{5.12}$$

where $\bar{w}$ denotes the w in substrategy $S(i, r_{p+1}, \bar{g}, v_p)$. By the inductive assumption, this point t' exists if $\sigma \in [2^{g(3)-r_{p+1}+2}, \frac{1}{2}]$. This holds because $r_p \geq h(0, i, r, g) = \lceil -\log \sigma \rceil + g(3) + 2$. Put $\delta(p) = \bar{w}(t')$ and $t_{p+1} = t_p + t'$. The inequality (5.12) implies that in the original game,

$$b^{r_p}_{r_{p+1}}(x_p y_p, t_{p+1}; V) \geq \gamma(i, \bar{g}) a(x_p y_p, t_{p+1}).$$

Therefore the conditions of Lemma 5.1(b) are satisfied. Hence, for any $v \in [2^{k-r+2}, \frac{1}{2}]$ there exists a t' with $w(t') \in [v, 2v]$ and

$$w^{r_0}_{r}(t'; V) \geq (\gamma(i, \bar{g}) + 2^{-k-3}) w(t') = \gamma(i+1, g) w(t').$$

Since $r_0 = f(i+1, r, g)$, this concludes the proof. □

6. Conclusion

The main result of this paper is that the monotonic complexity K is not equal to within an additive constant to the negative logarithm H of a priori probability, despite the fact that these two quantities are close to each other over broad regions. We must note that the time-bounded approximations of K and H may be quite different if the time bounds are small, since the proof of Lemma 4.1 uses exponential enumeration. The most natural open problem is, whether for a priori almost all sequences ω, the difference $K - H$ is bounded by a constant depending on ω. It is known (and proved in Section 2) that for a priori almost every ω, the difference $K - H$ grows slower than any unbounded recursive function.

Acknowledgment

I am grateful to R. Solovay whose careful reading improved the quality of the paper considerably, and to L.A. Levin for many illuminating conversations on the topic of this paper.

References

[1] Ya.M. Barzdin', The complexity of programs to determine whether natural numbers not greater than n belong to a recursively enumerable set, *Soviet Math. Dokl.* **9** (1968) 1251–1254.

[2] C.H. Bennett, On random and hard-to-describe numbers, *Research Report* RC 7483 (≠32272), IBM Research Center, Yorktown Heights NY 10598 (1979).

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

[4] P. Gács, On the symmetry of algorithmic information, *Soviet Math. Dokl.* **15** (1974) 1477–1480.

[5] P. Gács, Exact expressions for some randomness tests, *Z. Math. Logik. Grundl. Math.* **26** (1980).

[6] M. Gardner, The random number omega bids fair to hold mysteries of the universe, *Scientific American* **241**(5) (1979) 20–34.

[7] P.R. Halmos, *Measure Theory* (Van Nostrand, New York, 1950).

[8] A.N. Kolmogorov, Three approaches to the quantitative definition of information, *Problems Inform. Transmission* **1** (1965) 4–7.

[9] A.N. Kolmogorov, Logical basis for Information Theory and Probability Theory, *IEEE Trans. Inform. Theory* **14** (1968) 662–664.

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

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

[12] L.A. Levin, Laws of information conservation (nongrowth) and aspects of the foundations of probability theory, *Problems Inform. Transmission* **10**(3) (1974) 206–210.

[13] D.W. Loveland, A variant of the Kolmogorov concept of complexity, *Information and Control* **15** (1969) 510.

[14] P. Martin-Löf, The definition of random sequences, *Information and Control* **9** (1966) 602–619.

[15] T. Rado, On a simple source for non-computable functions, *Proc. Symposium Mathematical Theory of Automata*, MRI Symposium Series **XII** (Polytechnic Press, Brooklyn, 1962) 75–81.

[16] C.P. Schnorr, Process complexity and effective random tests, *J. Comput. System Sci.* **7** (1973) 376.

[17] R. Solomonoff, A formal theory of inductive inference, I., *Information and Control* **7** (1964) 1–22.

[18] R. Solovay, Unpublished manuscript (1976).