ON THE RELATION BETWEEN DESCRIPTIONAL COMPLEXITY AND ALGORITHMIC PROBABILITY

Péter Gács[1]
Computer Science Department
University of Rochester
Rochester, NY 14627

ABSTRACT: Several results in Algorithmic Information Theory establish upper bounds on the difference between descriptional complexity and the logarithm of "apriori 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 memory-allocation game between two 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 theories of inductive inference, descriptional complexity is a way to formalize "Occam's Razor"—the principle recommending the use of the simplest hypothesis among those consistent with the data. Another important principle known as "Bayes' Rule" assumes that a certain "apriori" probability distribution is defined over the set of possible states of the world and uses the conditional probability for inference. Algorithmic Information Theory originated from the recognition that descriptional complexity, if properly defined [Solomonoff 64, Kolmogorov 65], can be estimated by counting arguments and is a good approximation to the intuitive notion of entropy for individual objects. (For the exact elaboration of the analogy to entropy, see [Gacs 74, Chaitin 75].) Descriptional complexity can be successfully used for the definition of randomness [Martin-Löf 66, Kolmogorov 68, Levin 73]. Algorithmic Information Theory has a candidate for apriori probability which, when used in Bayes' Rule, gives satisfactory inferences over a wide range of situations. A simple but central result states that descriptional complexity is asymptotically equal to the absolute value of the logarithm of apriori probability [Solomonoff 64, Levin 70,73,74]. The theory 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 concerns the exactness of this relation.

Notation

Let Q denote the set of rational numbers, N the set of natural numbers, put $Z_2 = \{0,1\}$. For any set A, let $A^{(n)} = \cup_{i=0}^{n} A^i$ be the set of strings of length $\leq n$ with elements from A. Put $A^* = \cup_{n\geq 0} A^n$. 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 $\mathfrak{X} = N^N$, $\mathfrak{B} = Z_2^N$ for the sets of infinite strings of natural numbers and bits respectively. For $x, y \in N^*$, $x \subseteq y$ denotes that x is a prefix of y. For $(p_i, x_i) \in N^* \times N^*$, $(p_1, x_1) \subseteq (p_2, x_2)$ means $p_1 \subseteq p_2$, $x_1 \subseteq x_2$. The objects x and y are *compatible* if there is some z with $x \subseteq z$, $y \subseteq z$. For a string $x \in N^*$, put $x\mathfrak{X} = \{\omega \in \mathfrak{X} : x \subseteq \omega\}$. For a set $A \subseteq N^*$, put $A\mathfrak{X} = \cup_{x\in A} x\mathfrak{X}$. The operations $p\mathfrak{B}$, $A\mathfrak{B}$ are defined analogously. For a binary sequence $p \in \mathfrak{B}$, let $[p] \in [0,1]$ denote the real number associated with it in the binary number system. For $A \subseteq Z_2^*$, put $[A] = \{[\omega] : \omega \in A\mathfrak{B}\}$. 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 $E\mathfrak{X} \subseteq F\mathfrak{X}$ but the converse is not true. Let $l(x)$ denote the length of the sequence $x \in N^*$ and put $x^n = x_1 \cdots x_n$. 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 $f \preceq g$, $f \simeq g$ 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

[1] Part of this work appeared as a Stanford Technical Report while the author visited Stanford University in 1979. A full version is to appear in Theoretical Computer Science.

a luckier term than "descriptional complexity" both on account of its connotations and because there are too many different notions of "complexity". But since different quantities can claim to express the true meaning of algorithmic entropy, we abide by the term "descriptional complexity". It is worth defining descriptional complexity carefully. Several different definitions seem intuitively almost equally justified and are 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 [Levin 70,73]. We first tell how to *interpret* a description.

Definition 1.1 A r.e. (recursively enumerable) set $A \subseteq Z_2^* \times N^* \times N^*$ is called a *monotonic operator* (interpreter) if the following holds: whenever $(p_1, x_1, y_1), (p_2, x_2, y_2) \in A$ and (p_1, x_1) is compatible with (p_2, x_2), then y_1 is compatible with y_2. An interpreter defines a monotonic function by $A(p,x) = \sup\{ y \in N^* : \exists (p', x') \subseteq (p,x) \text{ with } (p', x', y) \in A \}$.

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 [Levin 74, Chaitin 75] 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 [Levin 73,74]). For $x, y \in N^\infty$,

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

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, $K_U(y \mid x) \leq K_A(y \mid x) + c_A$. Let us fix an optimal interpreter U, write $K(y \mid x) = K_U(y \mid x)$. The number $K(x) = K(x \mid \Lambda)$ is plainly called the complexity of x.

Typical orders of magnitude: for natural numbers n (sequences of length 1), $K(n) \preceq 2 \log n$, moreover, $K(n) \preceq \log n + K(\lfloor \log n \rfloor)$. This last estimate is sharp for most numbers $k \leq n$. For $x \in N^*$, if $x \subseteq U(p)$ and $l(p) = K(x)$ then p is called a *minimal description* of x. Let x^* denote the lexicographically smallest minimal description of x.

For $x \in N^*$, put $D_A(x) = \{ p \in N^* : \exists y \supseteq x \text{ with } (p, x) \in A \}$. Let us imagine 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. The work of such a machine can be represented by a monotonic operator with the special property that the set $\{ (p, x) : p \in (D_A(x))' \}$ is enumerable. The "process complexity" K' based on such machines and discovered in [Schnorr 73] independently for binary strings, may 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 [Martin-Löf 66]. In [Levin 73], monotonic complexity is used to characterize randomness. (These results are somewhat refined in [Gacs 80] where Martin-Löf's tests are expressed exactly by complexities.) But randomness is more immediately characterizable using another quantity, the *apriori probability*. It is worth trying to explain the role of its simple relation to apriori probability. Let $\pi = \pi_1, \pi_2, \ldots$ be an infinite sequence of identically distributed random variables with $\Pr[\pi_1 = 0] = \Pr[\pi_1 = 1] = 1/2$. The number

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

is the probability that our optimal machine produces x from a random input. In a more elementary notation,

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

Definition 1.3 The number $M(x)$ is called the *apriori probability* of x.

The apriori probability is not a probability measure over $\mathfrak{X}$ because on some $p \in \mathfrak{B}$, the sequence $U(p)$ does not have infinite length. Therefore some more definitions are needed. A nonnegative real function ν over N^* is a *semimeasure* if $\nu(\Lambda) \leq 1$ and for all $x \in N^*$,

$$\sum_{n \geq 0} \nu(xn) \leq \nu(x).$$

A semimeasure ν is a *measure* if equality holds here, a *probability* measure if also $\nu(\Lambda) = 1$. 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 above requirement 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 $\mu \leq \nu$. A measure can be uniquely extended to all Borel sets of $\mathfrak{X}$. The statement that a set S has *apriori probability* 0 means $\overline{M}(S) = 0$. The statement that a property holds for "apriori almost all ω" is used correspondingly. 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 $\mu(S) = \mu(S\mathfrak{X})$. The following properties, expressing some restricted monotonicity and additivity properties for semimeasures, are useful:

$$S_0 \le S_1 \Rightarrow \nu(S_0) \le \nu(S_1), \qquad \nu(\cup_n S_n) \le \sum_n \nu(S_n). \tag{1.2}$$

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 $g : N^* \times N \mapsto Q$ monotonically increasing in its second argument such that $f(x) = \lim_{t\to\infty} g(x,t)$ (g "generates" f). The apriori probability M is semicomputable because $M(x) = \lim_{t\to\infty} M_t(x)$ where $M_t(x) = \Pr[x \subseteq U_t(\pi)]$. 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. semimeasure 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:

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

For the generation of *computable* semimeasures, we need a pair of sequences: approximations from below and above. If the pair $T(\mathrm{pr}_1(e),x,t)$, $T(\mathrm{pr}_2(e),x,t)$ generates a computable semimeasure we denote this semimeasure by η_e. The computable semimeasures—just as the recursive functions—can not be enumerated. Therefore η_e does not always exist. Put $H(x) = -\log M(x)$. It follows from (1.1) that $H(x) \le K(x)$.

FACT 1.1

a) *For any computable semimeasure η_e,*

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

b) *ω is random with respect to a computable measure η_e iff there exists a constant c_ω such that for all n, $-\log \eta_e(\omega^n) \le H(\omega^n) + c_\omega$.*

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 [Levin 73] that $H(x) \simeq K(x)$. In the main result of this paper, we refute this conjecture.

A set $E \subseteq N^*$ is called *prefixfree* if its elements are incompatible. (Example: the set N^n of sequences of length n.) It is known (see [Levin 74]) that for any r.e. prefixfree set E a constant c_E exists such that we have $|H(x) - K(x)| \le c_E$ for all $x \in E$. It is enough to prove this for $E = N$, i.e. that

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

for natural numbers n; the rest follows by encoding. As a consequence, one can prove

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

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

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

—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*

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

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

Notice that for functions $g(n)$ semicomputable from above, $K \preceq g$ is equivalent to $\sum_n 2^{-g(n)} < \infty$.

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 apriori almost every ω, $K(\omega^n) - H(\omega^n)$ has an upper bound which is smaller than any unbounded recursive function.

The proof of Theorem 1.1 uses a two-person infinite game described in Section 3.

2. Information in largeness

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

$$\alpha(n) = \min_{n \le i} K(i) = \min\{\, l(p) : n \subseteq U(p) \,\}.$$

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 [Rado 62]).

These functions play an illuminating role in Algorithmic Information Theory. The following result is proved in [Barzdin' 68]. Let the infinite 0-1 sequence a be the characteristic function of a r.e. set. Let $a(n) = \langle a^{2^n}\rangle$ be the standard encoding of a 2^n-length prefix of a. Then $K(a(n) \mid n) \preceq n$ and for a suitable r.e. set given by $b(n)$ we have $K(b(n) \mid n) \simeq n$. What sort of information is stored in $b(n)$? It is algorithmically equivalent to a description of *large numbers.* Indeed: let $\{x(1), x(2), \dots\}$ be a recursive enumeration of the set B with characteristic function $b(n)$, $B_t = \{x(1), \dots, x(t)\}$ and $\rho(n) = \min\{t : b_t(n) = b(n)\}$ where b_t is the characteristic function of B_t. Then, given n, knowing $b(n)$ is the same as knowing any number larger than $\rho(n)$. On the other hand,

$$\alpha(\rho(n) \mid n) \simeq n,$$

i.e. $\rho(n)$, another "busy beaver" function, has "approximately" the same order of growth as the inverse of $\alpha(n)$. (We get the definition of $\alpha(n \mid x)$ by conditionalizing the definition of $\alpha(n)$.)

The length of $b(n)$ is 2^n, while it contains only n bits of information. Therefore it is not the shortest description of the large numbers it encodes. What do succinct definitions of large numbers look like? Consider the binary expansion Ω of the number $\sum_{n\in N} M(n)$. It is shown in [Chaitin 75] that $K(\Omega^n) \simeq n$ therefore it follows from Fact 1.1 b that Ω is random. It turns out that minimal definitions of large numbers are algorithmically equivalent to prefixes of Ω. 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

$$\alpha(\sigma(n) \mid n) \preceq n \preceq \alpha(\sigma(n)).$$

Ω_n can thus be considered a compressed form of $b(n)$ (which is 2^n long). Of course, the redundancy in $b(n)$ is not useless. The number $b(n)$ contains, in an easily accessible form, all significant information about the the results of computations performable in time $\alpha^{-1}(n)$. It is not surprising therefore that the computational complexity of any significant compression of $b(n)$ is nonrecursively large (see [Barzdin' 68]). For a popular exposition of this topic, see [Bennett 79, Gardner 79].

We will consider a natural "busy beaver" function associated with the apriori probability, which does not involve time-complexity at all. For any semimeasure ν, put

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

$s(n) = s(n; M)$. 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 $C(n) = \cup_{k\geq n} D(k)$. 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 [Solovay 76] that

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

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 $K(n) \preceq g(n)$.

NOTE Since we required monotonicity, $\log n + K(\lfloor \log n\rfloor) \preceq g(n)$ is also proved as $\log n + K(\lfloor \log n\rfloor)$ is $\simeq$ to the least monotonic upper bound on $K(n)$.

The proof of Theorem 2.1 uses a simple two-person game similar to the ones described in Section 3.

After looking at the probability of large numbers it is natural to ask how fast a sequence can increase if it is generated by a probabilistic Turing machine. The next two results of L.A.Levin have not been published before.

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

Once we are able to bound the growth of random sequences we also have an estimate 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 apriori almost all ω, there is a constant C such that*

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

3. Memory-allocation games

We describe a storage-allocation game between two players called User and Server.

Game 1

User sends requests at each step $t \in N$ for storage of different sorts of items. These sorts form a *hierarchical structure*: we identify them with N^*. Item 3 could mean "cars", item 30 "Fords", 311 "subcompact AMC cars", etc. Let the real number $a(x,t) \geq 0$ be the total quantity requested until time t from item x. Let $a(x,0) = 0$. At step t, $a(x,t)$ has rational values different from 0 only for finitely many $x \in N$, and $a(x,t)$ is obviously a semimeasure (see Section 2) for all t. The function $a(x,t)$ is monotonically increasing in t. Server has a store V which is the subset of the interval $[0,1]$. At each step t, Server allocates a set $B(x,t)$ (storage space) for item x which is a finite union of intervals. If x and y are incompatible then we must have $B(x,t) \cap B(y,t) = \emptyset$. Server is never allowed to reallocate (e.g. the store is a write-once memory). This is expressed by requiring that $B(x,t)$ be nondecreasing in t. Put $b(x,t) = \lambda(B(x,t))$, $c(x,t) = \max\{\, \lambda([p]) : [p] \subseteq B(x,t)\,\}$. Let $a(x)$, $B(x)$, $b(x)$ and $c(x)$ be the limits of $a(x,t)$, $B(x,t)$, $b(x,t)$ and $c(x,t)$ as $t \to \infty$. Server wants to maintain $\log \frac{a(x)}{c(x)} \leq g(l(x))$ for some function g. We can suppose that he satisfies

$$\log \frac{a(x,t)}{c(x,t)} \leq g(l(x)) \tag{3.1}$$

since otherwise User would just wait until (3.1) does not hold.

Theorem 3.1 *Suppose that $V = [0,1]$.*

a) Server has a recursive strategy to achieve

$$\log \frac{a(x)}{c(x)} \preceq \min\{K(l(x)), K(\lfloor -\log a(x) \rfloor)\}. \tag{3.2}$$

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

Proof of (1.6), (1.7) and Theorem 1.1:
First we prove (1.6) and (1.7). Put $a(x,t) = M_t(x)$. By Theorem 3.1 *a)*, there is a strategy $B(t,x)$ of Server with (3.2). Since $B(t,x)$ depends recursively on $a(t,x)$, the set $A = \{\,(p,x) : [p] \subseteq B(t,x)\,\}$ is a monotonic operator. Using (3.2) and the optimality of U,

$$\begin{aligned} K(x) &\preceq K_A(x) = -\log c(x) \\ &\preceq H(x) + \min\{K(l(x)), K(\lfloor H(x) \rfloor)\}. \end{aligned}$$

Now we prove Theorem 1.1. Let $g(n)$ be a function semicomputable from above with $2^{-g(n)} = \infty$, $g^k(x) = g(x) + k$. Put $B(x,t) = \bigcup\{\, [p] : x \sqsubseteq U_t(p)\,\}$. Theorem 3.1 *b)* says that for each k, there is a strategy $a^k(x,t)$ of User achieving $K(x) = -\log c(x) > -\log a^k(x) + g^k(l(x))$ for some x. We use a more constructive fact which also follows from the proof: namely that for some recursive function $f(k)$, we also can have $a^k = \psi_{f(k)}$. Hence (1.3) is applicable, and, using $K(f(k)) \preceq K(k)$, we get $K(x) \succeq H(x) + g(l(x)) + k - K(k)$ which implies $K(x) \succeq H(x) + g(l(x))$ for a k large enough. ∎

The positive part *a)* of Theorem 3.1 can be proved by the known techniques used for the proof of $K(n) \simeq H(n)$, after noticing that the set N^n of all sequences of a fixed length n is prefixfree. Server sets aside a store of size $O(M(n))$ and handles the sets N^n separately for each n. 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 gives the bound $K(\lfloor -\log a(x) \rfloor)$.

The proof of the negative part of Theorem 3.1 uses the idea of reservation. For any natural numbers $r \leq s$ and a set $E \subseteq V$ put

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

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. Put

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

$$B_r^s(x,t; V) = L_r^s(B(x,t); V - V'(x,t))$$

where $V'(x,t)$ is the union of $B(y,t)$ for all y incompatible with x. Put $w_r^s(t; V) = \lambda(W_r^s(t; V))$, $b_r^s(x,t; V) = \lambda(B_r^s(x,t; V))$. Notice that $w(t)$ is controlled by User but $W_r^s(t; V)$— 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 like an irrevocable allocation from the point of view of subsequent stages. To implement this idea we introduce a new game which serves as a recursion step in Game 1.

Game 2

The set of items is the set N^2 of sequences of length 2. The following additional parameters are given: an infinite nonincreasing sequence $r_0 \geq r_1 \geq \ldots$ of natural numbers, $r, k \in N$ with $k \leq r \leq r_i$, and a real number $\gamma > 0$. Put $\Gamma = \max\{1, \gamma\}$, $\sigma = \Gamma^{-1} 2^{-r-2}$. The rules are the same as for Game 1 and, additionally, the following. We have $a(x,t) = \sum_y a(xy,t)$ for all $x \in N$ and

$$a(x,t) \leq 2^{-r+k}. \tag{3.3}$$

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

$$a(x,t) = 2^{-r+k} \Rightarrow c(x,t) \geq 2^{-r} \tag{3.4}$$

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

$$b^{r_t}_{r_t+1}(xy,t+1) \geq \gamma a(xy,t+1). \tag{3.5}$$

The game ends at some time T. Rules (3.3) and (3.4) mean that the size of contiguous intervals requested in this game is always 2^{-r}.

Lemma 3.1 *In Game 2, User has a strategy with the following property. As long as Server is able to keep the rules for any $v \in [2^{k-r+2}, w(T)/2]$ there is a t' with $w(t') \in [v, 2v]$ and*

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

Proof of Lemma 3.1: 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 $y_t = \min\{\, y : B(x_t y, t) = \emptyset \,\}$. Put $F(t) = \bigcup_x B^r_r(x,t)$, $D(u,t) = F(u) \cap F(t)$, $d(t) = \lambda(F(t))$ and $C(t) = \bigcup_{u=1}^t B^{r_u-1}_{r_u}(x_u y_u, u)$. First, we prove that for all t,

$$w^{r_0}_r(t) \geq \gamma w(t) + d(t)/2. \tag{3.6}$$

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

$$\lambda(C(u) \cup D(u,t)) \geq \gamma w(t) + \lambda(D(u,t))/2 \tag{3.7}$$

which clearly implies (3.6). The inequality (3.7) 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 $x_u y_u$,

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

Therefore by (3.5), the left side of (3.7) increases by at least $b^{r_u}_{r_u+1}(x_u y_u, u+1) \geq \gamma\delta(u)$ while the right side increases exactly by $\gamma\delta(u)$, hence (3.7) 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 (3.7) 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 (3.7) holds for $u+1$. This proves (3.7) and hence (3.6). The inequality (3.6) gives

$$R(x_t,t) \leq 2^r d(t) \leq 2^{r+1}(w^{r_0}_r(t) - \gamma w(t)). \tag{3.8}$$

Inequality (3.8) 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.

Put

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

By the first inequality in (3.8), $E(t_0,t_1) = \{\, x : R(x,t) \leq m(t_0,t_1) \,\}$ is independent of t and $x_t y_t \in E(t_0,t_1)$ for all $t \in [t_0,t_1)$. Therefore

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

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

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

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

$$\begin{aligned} d(t') &= m2^{-r} \\ &\geq 2^{-k}(v - 2^{-r-1}) - 2^{-r} \\ &\geq 2^{-k-2}(4v - 2^{-r+1} - 2^{k-r+2}) \\ &\geq 2^{-k-2} w(t'). \end{aligned}$$

With (3.6), this completes the proof. ∎

Lemma 3.1 states that in the 2-level Game 2, User can force Server to devote to reservation significantly more area than the amount $\gamma w(t)$ required by the rules. A recursive application of the strategy of Lemma 3.1 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 3.2 *There is a recursive function $f(i, r, g)$ such that User has a strategy $S(i, r, g, V)$ in Game 1 with the following properties: for all i, $g : N \mapsto N$, $r \geq g(1)+3$, if Server satisfies (3.1) for all x with $l(x) \leq 2i+1$ then for all $v \in [2^{g(1)-r+2}, 1/2]$ there is a t' with $w(t') \in [v, 2v]$,*

$$w_r^{f(i,r,g)}(t';, V) \geq \gamma(i, g) w(t'). \qquad (3.9)$$

Proof of Theorem 3.1. 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 (3.1) then with $v = 1/4$ 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 3.1: We will prove the Lemma 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 *p-th 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 (3.1) guarantees (3.9). 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 $\overline{g}(n) = g(n+2)$, $\gamma = \gamma(i, \overline{g})$, $\Gamma = \max\{1, \gamma\}$, $k = g(1)$, $\sigma = \Gamma^{-1} 2^{-r-2}$ and $T = \lceil \sigma^{-1} \rceil$. We define the function f by the 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 p-th 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. Namely, the weight of some $x_p y_p$ increases by some $\delta(p) \in [\sigma, 2\sigma]$. The p-th macrostep itself is an application of strategy $S(i, r_p, \overline{g}, V_p)$ 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, $r_p = f(i, r_{p+1}, g)$ for all p with $w(t_{p+1}) \leq 1$. In $S(i, r_p, \overline{g}, V_p)$, the game is played until a point t' is reached with $\overline{w}(t') \in [\sigma, 2\sigma]$ and

$$\overline{w}_{r_{p+1}}^{r_p}(t'; V_p) \geq \gamma(i, \overline{g}) \overline{w}(t'), \qquad (3.10)$$

where $\overline{w}$ denotes the w in substrategy $S(i, r_p, \overline{g}, V_p)$. By the inductive assumption, this point t' exists if $\sigma \in [2^{g(3)-r_p+2}, 1/2]$. This holds because $r_p \geq h(0, i, r, g) = \lceil -\log \sigma \rceil + g(3) + 2$. Put $\delta(p) = \overline{w}(t')$, $t_{p+1} = t_p + t'$. Inequality (3.10) implies that in the original game,

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

Therefore the conditions of Lemma 3.1 *b)* are satisfied. Hence, for any $v \in [2^{k-r+2}, 1/2]$ there exists a t' with $w(t') \in [v, 2v]$ and

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

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

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

[Barzdin' 68] Ya.M.: The complexity of programs to determine whether natural numbers not greater than n belong to a recursively enumerable set. *Soviet Math. Doklady* **9** (1968) 1251-1254

[Bennett 79] C.H.: On random and hard-to-describe numbers. *Research Report* **RC 7483 (#32272)**, IBM Research Center, Yorktown Heights NY 10598

[Gacs 74] P.:On the symmetry of algorithmic information. *Soviet Math. Dokl.* **15** (1974) 1477-1480

[Gacs 80] P.:Exact expressions for some randomness tests. *Z. für Math. Log. u. Grdl. M.* **26** (1980)

[Gardner 79] M.: The random number omega bids fair to hold mysteries of the universe. *Scientific American* **241/5**, (Nov. 1979) 20-34

[Chaitin 75] G.: A theory of program-size formally identical to information theory. *J. ACM* **22** (1975) 329-340

[Kolmogorov 65] A.N.: Three approaches to the quantitative definition of information. *Problems of Inf. Transm.* **1** (1965) 4-7

[Kolmogorov 68] A.N.: Logical basis for Information Theory and Probability Theory. *IEEE Transactions on Information Theory* **IT-14** (1968) 662-664

[Levin 70] L.A., Zvonkin A.K.: 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

[Levin 73] L.A.: On the notion of a random sequence. *Soviet Math. Dokl.* **14/5** (1973) 1413-1416

[Levin 74] L.A.: Laws of information conservation (nongrowth) and aspects of the foundations of probability theory. *Problems of Inf. Transm.* **10/3** (1974) 206-210

[Martin-Löf 66] P.: The definition of random sequences. *Inf. and Contr.* **9** (1966) 602-619

[Rado 62] T.: On a simple source for non-computable functions. *Proc. Symp. Math. Theory of Automata* MRI Symposium Series XII, Polytechnic Press, Brooklyn (1962) 75-81

[Schnorr 73] C.P.: Process complexity and effective random tests *J. Comput. Syst. Sci.* **7** (1973) 376

[Solomonoff 64] R.: A formal theory of inductive inference. I.: *Inf. and Contr.* **7** (1964) 1-22

[Solovay 76] R.: Unpublished manuscript.