ALGORITHMIC ENTROPY OF SETS

Computers & Mathematics with Applications 2 (1976), pp. 233–245

Gregory J. Chaitin
IBM Thomas J. Watson Research Center
Yorktown Heights, NY 10598, U.S.A.

Abstract

In a previous paper a theory of program size formally identical to information theory was developed. The entropy of an individual finite object was defined to be the size in bits of the smallest program for calculating it. It was shown that this is $-\log_2$ of the probability that the object is obtained by means of a program whose successive bits are chosen by flipping an unbiased coin. Here a theory of the entropy of recursively enumerable sets of objects is proposed which includes the previous theory as the special case of sets having a single element. The primary concept in the generalized theory is the probability that a computing machine enumerates a given set when its program is manufactured by coin flipping. The entropy of a set is defined to be $-\log_2$ of this probability.

1. Introduction

In a classical paper on computability by probabilistic machines [1], de Leeuw *et al.* showed that if a machine with a random element can enumerate a specific set of natural numbers with positive probability, then there is a deterministic machine that also enumerates this set. We propose to throw further light on this matter by bringing into play the concepts of algorithmic information theory [2,3].

As in [3], we require a computing machine to read the successive bits of its program from a semi-infinite tape that has been filled with 0's and 1's by flipping an unbiased coin, and to decide by itself where to stop reading the program, for there is no endmarker. In [3] this convention has the important consequence that a program can be built up from subroutines by concatenating them.

In this paper we turn from finite computations to unending computations. The computer is used to enumerate a set of objects instead of a single one. An important difference between this paper and [3] is that here it is possible for the machine to read the entire program tape, so that in a sense infinite programs are permitted. However, following [1] it is better to think of these as cases in which a nondeterministic machine uses coin-flipping infinitely often.

Here, as in [3], we pick a universal computer that makes the probability of obtaining any given machine output as high as possible.

We are thus led to define three concepts: $P(A)$, the probability that the standard machine enumerates the set A, which may be called the algorithmic probability of the set A; $H(A)$, the entropy of the set A, which is $-\log_2$ of $P(A)$; and the amount of information that must be specified to enumerate A, denoted $I(A)$, which is the size in bits of the smallest program for A. In other words, $I(A)$ is the least number n such that for some program tape contents the standard machine enumerates the set A and in the process of doing so reads precisely n bits of the program tape.

One may also wish to use the standard machine to simultaneously enumerate two sets A and B, and this leads to the joint concepts $P(A,B)$, $H(A,B)$, and $I(A,B)$. In [3] programs could be concatenated, and this fact carries over here to programs that enumerate singleton sets (i.e. sets with a single element). What about arbitrary sets? Programs

that enumerate arbitrary sets can be merged by interweaving their bits in the order that they are read when running at the same time, that is, in parallel. This implies that the joint probability $P(A, B)$ is not less than the product of the individual probabilities $P(A)$ and $P(B)$, from which it is easy to show that H has all the formal properties of the entropy concept of classical information theory [4]. This also implies that $I(A, B)$ is not greater than the sum of $I(A)$ and $I(B)$.

The purpose of this paper is to propose this new approach and to determine what is the number of sets A that have probability $P(A)$ greater than 2^{-n}, in other words, that have entropy $H(A)$ less than n. It must be emphasized that we do not present a complete theory. For example, the relationship between $H(A)$ and $I(A)$ requires further study. In [3] we proved that the difference between $H(A)$ and $I(A)$ is bounded for singleton sets A, but we shall show that even for finite A this is no longer the case.

2. Definitions and Their Elementary Properties

The formal definition of computing machine that we use is the Turing machine. However, we have made a few changes in the standard definition [5, pp. 13–16].

Our Turing machines have three tapes: a program tape, a work tape and an output tape. The program tape is only infinite to the right. It can be read by the machine and it can be shifted to the left. Each square of the program tape contains a 0 or a 1. The program tape is initially positioned at its leftmost square. The work tape is infinite in both directions, can be read, written and erased, and can be shifted in either direction. Each of its squares may contain a blank, a 0, or a 1. Initially all squares are blank. The output tape is infinite in both directions and it can be written on and shifted to the left. Each square may contain a blank or a \$. Initially all squares are blank.

A Turing machine with n states, the first of which is its initial state, is defined in a table with $6n$ entries which is consulted each machine cycle. Each entry corresponds to one of the 6 possible contents of the

2 squares being read, and to one of the n states. All entries must be present, and each specifies an action to be performed and the next state. There are 8 possible actions: program tape left, output tape left, work tape left/right, write blank/0/1 on work tape and write \$ on output tape.

Each way of filling this $6 \times n$ table produces a different n-state Turing machine M. We imagine M to be equipped with a clock that starts with time 1 and advances one unit each machine cycle. We call a unit of time a quantum. Starting at its initial state M carries out an unending computation, in the course of which it may read all or part of the program tape. The output from this computation is a set of natural numbers A. n is in A iff a \$ is written by M on the output tape that is separated by exactly n blank squares from the previous \$ on the tape. The time at which M outputs n is defined to be the clock reading when two \$'s separated by n blanks appear on the output tape for the first time.

Let p be a finite binary sequence (henceforth *string*) or an infinite binary sequence (henceforth *sequence*). $M(p)$ denotes the set of natural numbers output *(enumerated)* by M with p as the contents of the program tape if p is a sequence, and with p written at the beginning of the program tape if p is a string. $M(p)$ is always defined if p is a sequence, but if p is a string and M reads beyond the end of p, then $M(p)$ is undefined. However, instead of saying that $M(p)$ is undefined, we shall say that $M(p)$ halts. Thus for any string p, $M(p)$ is either defined or halts. If $M(p)$ halts, the clock reading when M reads past the end of p is said to be the time at which $M(p)$ halts.

Definition.

- $P_M(A)$ is the probability that $M(p) = A$ if each bit of the sequence p is obtained by a separate toss of an unbiased coin. In other words, $P_M(A)$ is the probability that a program tape produced by coin flipping makes M enumerate A.

- $H_M(A) = -\log_2 P_M(A)$ $(= \infty$ if $P_M(A) = 0)$.

- $I_M(A)$ is the number of bits in the smallest string p such that $M(p) = A$ $(= \infty$ if no such p exists).

We now pick a particular universal Turing machine U having the ability to simulate any other machine as the standard one for use throughout this paper. U has the property that for each M there is a string $\%_M$ such that for all sequences p, $U(\%_M p) = M(p)$ and U reads exactly as much of p as M does. To be more precise $\%_M = 0^g 1$, where g is the Gödel number for M. That is to say, g is the position of M in a standard list of all possible Turing machine defining tables.

Definition.

- $P(A) = P_U(A)$ is the algorithmic probability of the set A.
- $H(A) = H_U(A)$ is the algorithmic entropy of the set A.
- $I(A) = I_U(A)$ is the algorithmic information of the set A.

The qualification "algorithmic" is usually omitted below.

We say that a string or sequence p is a program for A if $U(p) = A$. If $U(p) = A$ and p is a string of $I(A)$ bits, then p is said to be a minimal-size program for A. The recursively enumerable (r.e.) sets are defined to be those sets of natural numbers A for which $I(A) < \infty$. (This is equivalent to the standard definition [5, p. 58].) As there are nondenumerably many sets of natural numbers and only denumerably many r.e. sets, most A have $I(A) = \infty$.

The following theorem, whose proof is immediate, shows why U is a good machine to use. First some notation must be explained. $f(x) \preceq g(x)$ means that $\exists c\ \forall x\ f(x) \leq cg(x)$. $f(x) \succeq g(x)$ means that $g(x) \preceq f(x)$. And $f(x) \asymp g(x)$ means that $f(x) \preceq g(x)$ and $f(x) \succeq g(x)$. $O(f(x))$ denotes an $F(x)$ with the property that there are constants c_1 and c_2 such that for all x, $|F(x)| \leq |c_1 f(x)| + |c_2|$, where $f(x)$ is to be replaced by 0 if it is undefined for a particular value of x.

Theorem 1. $P(A) \succeq P_M(A)$, $H(A) \leq H_M(A) + O(1)$, and $I(A) \leq I_M(A) + O(1)$.

Definition.

- A join $B = \{2n : n \in A\} \cup \{2n+1 : n \in B\}$. [5, pp. 81, 168]. Enumerating A join B is equivalent to simultaneously enumerating A and B.

- $P(A,B) = P(A \text{ join } B)$ (joint probability)
- $H(A,B) = H(A \text{ join } B)$ (joint entropy)
- $I(A,B) = I(A \text{ join } B)$ (joint information)
- $P(A/B) = P(A,B)/P(B)$ (conditional probability)
- $H(A/B) = -\log_2 P(A/B) = H(A,B) - H(B)$ (conditional entropy)
- $P(A:B) = P(A)P(B)/P(A,B)$ (mutual probability)
- $H(A:B) = -\log_2 P(A:B) = H(A) + H(B) - H(A,B)$ (mutual entropy).

Theorem 2.

(a) $P(A) \geq 2^{I(A)}$

(b) $H(A) \leq I(A)$

(c) For singleton A, $H(A) = I(A) + O(1)$.

(d) $H(A) < \infty$ implies $I(A) < \infty$.

Proof. (a) and (b) are immediate; (c) is Theorem 3.5(b) [3]; (d) follows from Theorem 2 [1].

Theorem 3.

(a) $P(A,B) \asymp P(B,A)$

(b) $P(A,A) \asymp P(A)$

(c) $P(A,\emptyset) \asymp P(A)$

(d) $P(A/A) \asymp 1$

(e) $P(A/\emptyset) \asymp P(A)$

(f) $P(A,B) \succeq P(A)P(B)$

(g) $P(A/B) \succeq P(A)$

(h) $\sum_A P(A,B) \asymp P(B)$

(i) $P(A,B) \preceq P(B)$

(j) $\sum_A P(A/B) \asymp 1$.

Proof. The proof is straightforward. For example, (f) was shown in Section 1. And (h) follows from the fact that there is a $\% = 0^g 1$ such that $n \in U(\%p)$ iff $2n+1 \in U(p)$. Thus $P(B) \geq 2^{-|\%|} \sum_A P(A,B)$, which taken together with (b) yields (h). Here, and henceforth, the absolute value $|s|$ of a string s signifies the number of bits in s.

The remainder of the proof is omitted.

Theorem 4.

(a) $H(A,B) = H(B,A) + O(1)$

(b) $H(A,A) = H(A) + O(1)$

(c) $H(A,\emptyset) = H(A) + O(1)$

(d) $H(A/A) = O(1)$

(e) $H(A/\emptyset) = H(A) + O(1)$

(f) $H(A,B) \leq H(A) + H(B) + O(1)$

(g) $H(A/B) \leq H(A) + O(1)$

(h) $H(A) \leq H(A,B) + O(1)$

(i) $H(A:\emptyset) = O(1)$

(j) $H(A:A) = H(A) + O(1)$

(k) $H(A:B) = H(B:A) + O(1)$

(l) $H(A:B) = H(A) - H(A/B) + O(1)$.

Theorem 5.

(a) $I(A,B) = I(B,A) + O(1)$

(b) $I(A,A) = I(A) + O(1)$

(c) $I(A,B) \leq I(A) + I(B) + O(1)$

(d) $I(A) \leq I(A,B) + O(1)$

(e) $I(A) = I(A, \{n : n < I(A)\}) + O(1)$.

The proofs of Theorems 4 and 5 are straightforward and are omitted.

2′. The Oracle Machine U′

In order to study P, H, and I, which are defined in terms of U, we shall actually need to study a more powerful machine called U', which, unlike U, could never actually be built. U' is almost identical to U, but it cannot be built because it contains one additional feature, an oracle that gives U' *yes/no* answers to specific questions of the form "Does $U(p)$ halt?" U' can ask the oracle such questions whenever it likes. An oracle is needed because of a famous theorem on the undecidability of the halting problem [5, pp. 24–26], which states that there is no algorithm for answering these questions. U' is a special case of the general concept of relative recursiveness [5, pp. 128–134].

As a guide to intuition it should be stated that the properties of U' are precisely analogous to those of U; one simply imagines a universe exactly like ours except that sealed oracle boxes can be computer subsystems. We now indicate how to modify Section 2 so that it applies to U' instead of U.

One begins by allowing an oracle machine M to indicate in each entry of its table one of 9 possible actions (before there were 8). The new possibility is to ask the oracle if the string s currently being read on the work tape has the property that $U(s)$ halts. In response the oracle instantly writes a 1 on the work tape if the answer is *yes* and writes a 0 if the answer is *no.*

After defining an arbitrary oracle machine M, and P'_M, H'_M, and I'_M, one then defines the standard oracle machine U' which can simulate any M. The next step is to define $P'(A)$, $H'(A)$, and $I'(A)$, which are the probability, entropy, and information of the set A relative to the halting problem. Furthermore, p is said to be an oracle program for A if $U'(p) = A$, and a minimal-size one if in addition $|p| = I'(A)$. Then A

is defined to be r.e. in the halting problem if $I'(A) < \infty$. One sees as before that P' is maximal and H' and I' are minimal, and then defines the corresponding joint, conditional, and mutual concepts. Lastly one formulates the proves the corresponding Theorems $2'$, $3'$, $4'$, and $5'$.

Theorem 6. $P'(A) \succeq P(A)$, $H'(A) \leq H(A) + O(1)$, and $I'(A) \leq I(A) + O(1)$.

Proof. There is a $\% = 0^g1$ such that for all sequences p, $U'(\%p) = U(p)$ and U' reads precisely as much of p as U does.

3. Summary and Discussion of Results

The remainder of this paper is devoted to counting the number of sets A of different kinds having information $I(A)$ less than n and having entropy $H(A)$ less than n. The kinds of A we shall consider are: singleton sets, consecutive sets, finite sets, cofinite sets, and arbitrary sets. A is consecutive if it is finite and $n+1 \in A$ implies $n \in A$. A is cofinite if it contains all but finitely many natural numbers.

The following 4 pairs of estimates will be demonstrated in this paper. The first pair is due to Solovay [6]. $\#X$ denotes the cardinality of X. S_n denotes the Singleton set $\{n\}$. C_n denotes the Consecutive set $\{k : k < n\}$.

$$\begin{aligned} \log_2 \#\{\text{singleton } A : I(A) < n\} &= n - I(S_n) + O(1). \\ \log_2 \#\{\text{singleton } A : H(A) < n\} &= n - I(S_n) + O(1). \end{aligned}$$

$$\begin{aligned} \log_2 \#\{\text{consecutive } A : I(A) < n\} &= n - I(C_n) + O(\log I(C_n)). \\ \log_2 \#\{A : I(A) < n\} &= n - I(C_n) + O(\log I(C_n)). \end{aligned}$$

$$\begin{aligned} \log_2 \#\{\text{consecutive } A : H(A) < n\} &= n - I'(S_n) + O(1). \\ \log_2 \#\{\text{finite } A : H(A) < n\} &= n - I'(S_n) + O(1). \end{aligned}$$

$$\begin{aligned} \log_2 \#\{\text{cofinite } A : H(A) < n\} &= n - I'(C_n) + O(\log I'(C_n)). \\ \log_2 \#\{A : H(A) < n\} &= n - I'(C_n) + O(\log I'(C_n)). \end{aligned}$$

These estimates are expressed in terms of $I(S_n)$, $I(C_n)$, $I'(S_n)$ and $I'(C_n)$. These quantities are variations on a theme: specifying the natural number n in a more or less constructive manner. $I(S_n)$ is the number of bits of information needed to directly calculate n. $I(C_n)$ is the number of bits of information needed to obtain n in the limit from below. $I'(S_n)$ is the number of bits of information needed to directly calculate n using an oracle for the halting problem. And $I'(C_n)$ is the number of bits of information needed to obtain n in the limit from below using an oracle for the halting problem. The following theorem, whose straightforward proof is omitted, gives some facts about these quantities and the relationship between them.

Theorem 7. $I(S_n), I(C_n), I'(S_n)$ and $I'(C_n)$

(a) All four quantities vary smoothly. For example, $|I(S_n)-I(S_m)| \leq O(\log|n-m|)$, and the same inequality holds for the other three quantities.

(b) For most n all four quantities are $\log_2 n + O(\log\log n)$. Such n are said to be *random* because they are specified by table look-up without real computation.

(c) The four ways of specifying n are increasingly indirect:

$$\begin{aligned} I'(C_n) &\leq I'(S_n) + O(1), \\ I'(S_n) &\leq I(C_n) + O(1), \text{ and} \\ I(C_n) &\leq I(S_n) + O(1). \end{aligned}$$

(d) Occasionally n is random with respect to one kind of specification, but has a great deal of pattern and its description can be considerably condensed if more indirect means of specification are allowed. For example, the least $n \geq 2^k$ such that $I(S_n) \geq k$ has the following properties: $n < 2^{k+1}$, $I(S_n) = k + O(\log k)$ and $I(C_n) \leq \log_2 k + O(\log\log k)$. This relationship between $I(S_n)$ and $I(C_n)$ also holds for $I(C_n)$ and $I'(S_n)$, and for $I'(S_n)$ and $I'(C_n)$.

We see from Theorem 7(b) that all 4 pairs of estimates for $\log_2 \#_n$ are usually $n - \log_2 n + O(\log\log n)$ and thus close to each other. But

Theorem 7(c) shows that the 4 pairs are shown above in what is essentially ascending numerical order. In fact, by Theorem 7(d), for each k there is an n such that $k = \log_2 n + O(1)$ and one pair of estimates is that

$$\log_2 \#_n = n - \log_2 n + O(\log\log n)$$

while the next pair is that

$$\log_2 \#_n = n - (\text{a quantity} \leq \log_2 \log_2 n) + O(\log\log\log n).$$

Hence each pair of cardinalities can be an arbitrarily small fraction of the next pair.

Having examined the comparative magnitude of these cardinalities, we obtain two corollaries.

As was pointed out in Theorem 2(c), for singleton sets $I(A) = H(A) + O(1)$. Suppose consecutive sets also had this property. Then using the fifth estimate and Theorem 7(a) one would immediately conclude that $\#\{\text{consecutive } A : I(A) < n\} \asymp \#\{\text{consecutive } A : H(A) < n\}$. But we have seen that the first of these cardinalities can be an arbitrarily small fraction of the second one. This contradiction shows that consecutive sets do not have the property that $I(A) = H(A) + O(1)$. Nevertheless, in Section 5 it is shown that these sets do have the property that $I(A) = H(A) + O(\log H(A))$. Further research is needed to clarify the relationship between $I(A)$ and $H(A)$ for A that are neither singleton nor consecutive.

It is natural to ask what is the relationship between the probabilities of sets and the probabilities of their unions, intersections, and complements. $P(A \cup B) \succeq P(A, B) \succeq P(A)P(B)$, and the same inequality holds for $P(A \cap B)$. But is $P(\overline{A}) \succeq P(A)$? If this were the case, since the complement of a cofinite set is finite, using the sixth estimate and Theorem 7(a) it would immediately follow that $\#\{\text{finite } A : H(A) < n\}$ is $\succeq \#\{\text{cofinite } A : H(A) < n\}$. But we have seen that the first of these cardinalities can be an arbitrarily small fraction of the second. Hence it is not true that $P(\overline{A}) \succeq P(A)$. However in Section 7 it is shown that $P'(\overline{A}) \succeq P(A)$.

Corollary 1.

(a) For consecutive A it is not true that $I(A) = H(A) + O(1)$.

(b) For cofinite A it is not true that $P(\overline{A}) \succeq P(A)$.

4. The Estimates Involving Singleton Sets

The following theorem and its proof are due to Solovay [6], who formulated them in a string-entropy setting.

Definition. Consider a program p for a singleton set A. The bits of p which have not been read by U by the time the element of A is output are said to be *superfluous.*

Theorem 8.

(a) $\log_2 \#\{\text{singleton } A : I(A) < n\} = n - I(S_n) + O(1)$.

(b) $\log_2 \#\{\text{singleton } A : H(A) < n\} = n - I(S_n) + O(1)$.

Proof. (b) follows immediately from (a) by using Theorems 2(c) and 7(a). To prove (a) we break it up into two assertions: an upper bound on $\log_2 \#$, and a lower bound.

Let us start by explaining how to *mend* a minimal-size program for a singleton set. The program is mended by replacing each of its superfluous bits by a 0 and adding an endmarker 1 bit.

There is an $\% = 0^g 1$ such that if p is a mended minimal-size program for S_j, then $U(\%p) = \{|p| - 1\} = \{I(S_j)\}$. $\%$ accomplishes this by instructing U to execute p in a special way: when U would normally output the first number, it instead immediately advances the program tape to the endmarker 1 bit, outputs the amount of tape that has been read, and goes to sleep.

The crux of the matter is that with this $\%$

$$P(S_m) \geq \#\{j : I(S_j) = m\} 2^{-|\%|-m-1},$$

and so

$$\#\{j : I(S_j) = m\} \preceq P(S_m) 2^m.$$

Substituting $n-k$ for m and summing over all k from 1 to n, we obtain

$$\#\{j : I(S_j) < n\} \preceq P(S_n) 2^n \sum_{k=1}^{n} (P(S_{n-k})/P(S_n)) 2^{-k}.$$

It is easy to see that $P(S_n) \succeq P(S_k) P(S_{n-k})$ and so $P(S_{n-k})/P(S_n) \preceq 1/P(S_k) \preceq k^2$. Hence the above summation is $\preceq$

$$\sum_{k=1}^{n} k^2 2^{-k},$$

which converges for $n = \infty$. Thus

$$\#\{j : I(S_j) < n\} \preceq P(S_n)2^n.$$

Taking logarithms of both sides and using Theorem 2(c) we finally obtain

$$\log_2 \#\{j : I(S_j) < n\} \leq n - I(S_n) + O(1).$$

This upper bound is the first half of the proof of (a). To complete the proof we now obtain the corresponding lower bound.

There is a $\% = 0^g1$ with the following property. Concatenate $\%$, a minimal-size program p for S_n with all superfluous bits deleted, and an arbitrary string s that brings the total number of bits up to $n-1$. $\%$ is chosen so that $U(\%p) = S_k$, where k has the property that s is a binary numeral for it.

$\%$ instructs U to proceed as follows with the rest of its program, which consists of the subroutine p followed by $n-1-|\%p|$ bits of data s. First U executes p to obtain n. Then U calculates the size of s, reads s, converts s to a natural number k, outputs k, and goes to sleep.

The reason for considering this $\%$ is that $\log_2$ of the number of possible choices for s is $|s| = |\%ps| - |\%p| = n - 1 - |\%| - |p| \geq n - 1 - |\%| - I(S_n)$. And each choice of s yields a different singleton set $S_k = U(\%ps)$ such that $I(S_k) \leq |\%ps| = n - 1$. Hence

$$\log_2 \#\{k : I(S_k) < n\} \geq n - 1 - |\%| - I(S_n) = n - I(S_n) + O(1).$$

The proof of (a), and thus of (b), is now complete.

Theorem $8'$**.**

(a) $\log_2 \#\{\text{singleton } A : I'(A) < n\} = n - I'(S_n) + O(1).$

(b) $\log_2 \#\{\text{singleton } A : H'(A) < n\} = n - I'(S_n) + O(1).$

Proof. Imagine that the proofs of Theorem 8 and its auxiliary theorems refer to U' instead of U.

5. The Remaining Estimates Involving I(A)

Definition.

- $Q(n) = \sum P(A)$ $(\#A < n)$ is the probability that a set has less than n elements.
- $Q(n)^t$ is the probability that with a program tape produced by coin flipping, U outputs less than n different numbers by time t. Note that $Q(n)^t$ can be calculated from n and t, and is a rational number of the form $k/2^t$ because U can read at most t bits of program by time t.

Lemma 1.

(a) $Q(0) = 0$, $Q(n) \leq Q(n+1)$, $\lim_{n\to\infty} Q(n) < 1$.

(b) $Q(0)^t = 0$, $Q(n)^t \leq Q(n+1)^t$, $\lim_{n\to\infty} Q(n)^t = 1$.

(c) For $n > 0$, $Q(n)^0 = 1$, $Q(n)^t \geq Q(n)^{t+1}$, $\lim_{t\to\infty} Q(n)^t = Q(n)$.

(d) If A is finite, then $Q(\#A + 1) - Q(\#A) \geq P(A)$.

Theorem 9. If A is consecutive and $P(A) > 2^{-n}$, then $I(A) \leq n + I(C_n) + O(1)$.

Proof. There is a $\% = 0^g1$ with the following property. After reading $\%$, U expects to find on its program tape a string of length $I(C_n) + n$ which consists of a minimal-size program p for C_n appropriately merged with the binary expansion of a rational number $x = j/2^n$ $(0 \leq j < 2^n)$. In parallel U executes p to obtain C_n, reads x, and outputs a consecutive set. This is done in stages.

U begins stage t $(t = 1, 2, 3, \ldots)$ by simulating one more time quantum of the computation that yields C_n. During this simulation, whenever it is necessary to read another bit of the program U supplies this bit by reading the next square of the actual program tape. And whenever the simulated computation produces a new output (this will occur n times), U instead takes this as a signal to read the next bit of x from the program tape. Let x_t denote the value of x based on what U has read from its program tape by stage t. Note that $0 \leq x_t \leq x_{t+1}$ and $\lim_{t\to\infty} x_t = x < 1$.

In the remaining portion of stage t U does the following. It calculates $Q(k)^t$ for $k = 0, 1, 2, \ldots$ until $Q(k)^t = 1$. Then it determines m_t which is the greatest value of k for which $Q(k)^t \leq x_t$. Note that

since $Q(0)^t = 0$ there is always such a k. Also, since $Q(k)^t$ is monotone decreasing in t, and x_t is monotone increasing, it follows that m_t is also monotone increasing in t. Finally U outputs the m_t natural numbers less than m_t, and proceeds to stage $t+1$.

This concludes the description of the instructions incorporated in %. % is now used to prove the theorem by showing that if A is consecutive and $P(A) > 2^{-n}$, then $I(A) \leq n + I(C_n) + |\%|$.

As pointed out in the lemma, $Q(\#A+1) - Q(\#A) \geq P(A) > 2^{-n}$. It follows that the open interval of real numbers between $Q(\#A)$ and $Q(\#A+1)$ contains a rational number x of the form $j/2^n$ $(0 \leq j < 2^n)$. It is not difficult to see that one obtains a program for A that is $|\%|+I(C_n)+n$ bits long by concatenating % and the result of merging in an appropriate fashion a minimal-size program for C_n with the binary expansion of x. Hence $I(A) \leq n + I(C_n) + |\%| = n + I(C_n) + O(1)$.

Theorem 10. If A is consecutive $I(A) = H(A) + O(\log H(A))$ and $H(A) = I(A) + O(\log I(A))$.

Proof. Consider a consecutive set A. By Theorem 7(a), $I(C_n) = O(\log n)$. Restating Theorem 9, if $H(A) < n$ then $I(A) \leq n + I(C_n) + O(1) = n + O(\log n)$. Taking $n = H(A) + 1$, we see that $I(A) \leq H(A) + O(\log H(A))$. Moreover, $H(A) \leq I(A)$ (Theorem 2(b)). Hence $I(A) = H(A) + O(\log H(A))$, and thus $I(A) = H(A) + O(\log I(A))$.

Theorem 11. $\log_2 \#\{A : I(A) < n\} \leq n - I(C_n) + O(\log I(C_n))$.

Proof. There is a $\% = 0^g 1$ with the following property. Let p be an arbitrary sequence, and suppose that U reads precisely m bits of the program p. Then $U(\%p) = C_m$. % accomplishes this by instructing U to execute p in a special way: normal output is replaced by a continually updated indication of how many bits of program have been read.

The crux of the matter is that with this %

$$P(C_m) \geq \#\{A : I(A) = m\} 2^{-|\%|-m},$$

and so

$$\#\{A : I(A) = m\} \preceq P(C_m) 2^m.$$

Replacing m by $n-k$ and summing over all k from 1 to n, we obtain

$$\#\{A : I(A) < n\} \preceq P(C_n) 2^n \sum_{k=1}^{n} (P(C_{n-k})/P(C_n)) 2^{-k}.$$

It is easy to see that $P(C_n) \succeq P(C_k)P(C_{n-k})$ and so $P(C_{n-k})/P(C_n) \preceq 1/P(C_k) \preceq k^2$. Hence the above summation is $\preceq$

$$\sum_{k=1}^{n} k^2 2^{-k},$$

which converges for $n = \infty$. Thus

$$\#\{A : I(A) < n\} \preceq P(C_n)2^n.$$

Taking logarithms of both sides and using

$$\log_2 P(C_n) = -I(C_n) + O(\log I(C_n))$$

(Theorem 10), we finally obtain

$$\log_2 \#\{A : I(A) < n\} \leq n - I(C_n) + O(\log I(C_n)).$$

Theorem 12.

$$\log_2 \#\{\text{consecutive } A : I(A) < n\} \geq n - I(C_n) + O(\log I(C_n)).$$

Proof. There is a $\% = 0^g 1$ that is used in the following manner. Concatenate these strings: $\%$, a minimal-size program for $\{I(C_n)\}$ with all superfluous bits deleted, a minimal-size program for C_n, and an arbitrary string s of size sufficient to bring the total number of bits up to $n-1$. Call the resulting $(n-1)$-bit string p. Note that s is at least $n - 1 - |\%| - I(\{I(C_n)\}) - I(C_n)$ bits long. Hence $\log_2$ of the number of possible choices for s is, taking Theorem 7(a) into account, at least $n - I(C_n) + O(\log I(C_n))$.

$\%$ instructs U to proceed as follows with the rest of p, which consists of two subroutines and the data s. First U executes the first subroutine in order to calculate the size of the second subroutine and know where s begins. Then U executes the second subroutine, and uses each new number output by it as a signal to read another bit of the data s. Note that U will never know when it has finished reading s. As U reads the string s, it interprets s as the reversal of the binary numeral for a natural number m. And U contrives to enumerate the set C_m by outputting 2^k consecutive natural numbers each time the kth bit of s that is read is a 1.

To recapitulate, for each choice of s one obtains an $(n-1)$-bit program p for a different consecutive set (in fact, the set C_m, where s is the reversal of a binary numeral for m). In as much as $\log_2$ of the number of possible choices for s was shown to be at least $n - I(C_n) + O(\log I(C_n))$, we conclude that $\log_2 \#\{\text{consecutive } A : I(A) < n\}$ is $\geq n - I(C_n) + O(\log I(C_n))$.

Theorem 13.

(a) $\log_2 \#\{\text{consecutive } A : I(A) < n\} = n - I(C_n) + O(\log I(C_n))$.

(b) $\log_2 \#\{A : I(A) < n\} = n - I(C_n) + O(\log I(C_n))$.

Proof. Since $\#\{\text{consecutive } A : I(A) < n\} \leq \#\{A : I(A) < n\}$, this follows immediately from Theorems 12 and 11.

Theorem 13′.

(a) $\log_2 \#\{\text{consecutive } A : I'(A) < n\} = n - I'(C_n) + O(\log I'(C_n))$.

(b) $\log_2 \#\{A : I'(A) < n\} = n - I'(C_n) + O(\log I'(C_n))$.

Proof. Imagine that the proofs of Theorem 13 and its auxiliary theorems refer to U' instead of U.

6. The Remaining Lower Bounds

In this section we construct many consecutive sets and cofinite sets with probability greater than 2^{-n}. To do this, computations using an oracle for the halting problem are simulated using a fake oracle that answers that $U(p)$ halts iff it does so within time t. As t goes to infinity, any finite set of questions will eventually be answered correctly by the fake oracle. This *simulation in the limit* % is used to: (a) take any n-bit oracle program p for a singleton set and construct from it a consecutive set $U(\%px)$ with probability greater than or equal to $2^{-|\%|-n}$, and (b) take any n-bit oracle program p for a consecutive set and construct from it a cofinite set $U(\%px)$ with probability greater than or equal to $2^{-|\%|-n}$.

The critical feature of the simulation in the limit that accomplishes (a) and (b) can best be explained in terms of two notions: *harmless*

overshoot and *erasure.* The crux of the matter is that although in the limit the fake oracle realizes its mistakes and changes its mind, U may already have read beyond p into x. This is called overshoot, and could make the probability of the constructed set fall far below $2^{-|\%|-n}$. But the construction process contrives to make overshoot harmless by eventually forgetting bits in x and by erasing its mistakes. In case (a) erasure is accomplished by moving the end of the consecutive set. In case (b) erasure is accomplished by filling in holes that were left in the cofinite set. As a result bits in x do not affect which set is enumerated; they can only affect the time at which its elements are output.

Lemma 2. With our Turing machine model, if k is output at time $\leq t$, then $k < t < 2^t$.

Theorem 14. There is a $\% = 0^g1$ with the following property. Suppose the string p is an oracle program for S_k. Let t_1 be the time at which k is output. Consider the finite set of questions that are asked to the oracle during these t_1 time quanta. Let t_2 be the maximum time taken to halt by any program that the oracle is asked about. ($t_2 = 0$ if none of them halt or if no questions are asked.) Finally, let $t = \max t_1, t_2$. Then for all sequences x, $\%px$ is a program for the set C_l, where $l = 2^t + k$. By the lemma k can be recovered from l.

Proof. $\%$ instructs U to act as follows on px. Initially U sets $i = 0$. Then U works in stages. At stage t ($t = 1, 2, 3, \ldots$) U simulates t time quanta of the computation $U'(px)$, but truncates the simulation immediately if U' outputs a number. U fakes the halting-problem oracle used by U' by answering that a program halts iff it takes $\leq t$ time quanta to do so. Did an output k occur during the simulated computation? If not, nothing more is done at this stage. If so, U does the following. First it sets $i = i + 1$. Let L_i be the chronological list of *yes/no* answers given by the fake oracle during the simulation. U checks whether $i = 1$ or $L_{i-1} \neq L_i$. (Note that $L_{i-1} = L_i$ iff the same questions were asked in the same order and all the answers are the same.) If $i > 1$ and $L_{i-1} = L_i$, U does nothing at this stage. If $i = 1$ or $L_{i-1} \neq L_i$, U outputs all natural numbers less than $2^t + k$, and proceeds to stage $t + 1$.

It is not difficult to see that this $\%$ proves the theorem.

Theorem 15. $\log_2 \#\{\text{consecutive } A : H(A) < n\} \geq n - I'(S_n) + O(1)$.

Proof. By Theorem 14, $c = |\%|$ has the property that for each singleton set S_k such that $I'(S_k) < n - c$ there is a different l such that $P(C_l) > 2^{-n}$. Hence in view of Theorems 8(a)′ and 7(a)

$$\begin{aligned} & \log_2 \#\{\text{consecutive } A : H(A) < n\} \\ \geq\ & \log_2 \#\{\text{singleton } A : I'(A) < n - c\} \\ \geq\ & n - c - I'(S_{n-c}) + O(1) \\ =\ & n - I'(S_n) + O(1). \end{aligned}$$

Theorem 16. There is a $\% = 0^g 1$ with the following property. Suppose the string p is an oracle program for the finite set A. For each $k \in A$, let t_k^1 be the time at which it is output. Also, let t_k^2 be the maximum time taken to halt by any program that the oracle is asked about during these t_k^1 time quanta. Finally, let $t_k = \max t_k^1, t_k^2$, and $l_k = 2^{t_k} + k$. Then for all sequences x, $\%px$ is a program for the cofinite set $B =$ all natural numbers not of the form l_k ($k \in A$). By the lemma each k in A can be recovered from the corresponding l_k.

Proof. % instructs U to act as follows on px in order to produce B. U works in stages. At stage t ($t = 1, 2, 3, \ldots$) U simulates t time quanta of the computation $U'(px)$. U fakes the halting-problem oracle used by U' by answering that a program halts iff it takes $\leq t$ time quanta to do so. While simulating $U'(px)$, U notes the time at which each output k occurs. U also keeps track of the latest stage at which a change occurred in the chronological list of *yes/no* answers given by the fake oracle during the simulation before k is output. Thus at stage t there are current estimates for t_k^1, for t_k^2, and for $t_k = \max t_k^1, t_k^2$, for each k that currently seems to be in $U'(px)$. As t goes to infinity these estimates will attain the true values for $k \in A$, and will not exist or will go to infinity for $k \notin A$.

Meanwhile U enumerates B. That part of B output by stage t consists precisely of all natural numbers less than 2^{t+1} that are not of the form $2^{t_k} + k$, for any k in the current approximation to $U'(px)$. Here t_k denotes the current estimate for the value of t_k.

It is not difficult to see that this % proves the theorem.

Theorem 17.

$$\log_2 \#\{\text{cofinite } A : H(A) < n\} \geq n - I'(C_n) + O(\log I'(C_n)).$$

Proof. By Theorem 16, $c = |\%|$ has the property that for each consecutive set A such that $I'(A) < n - c$ there is a different cofinite set B such that $P(B) > 2^{-n}$. Hence in view of Theorems 13(a)′ and 7(a)

$$\begin{aligned} & \log_2 \#\{\text{cofinite } B : H(B) < n\} \\ \geq\ & \log_2 \#\{\text{consecutive } A : I'(A) < n - c\} \\ \geq\ & n - c - I'(C_{n-c}) + O(\log I'(C_{n-c})) \\ \geq\ & n - I'(C_n) + O(\log I'(C_n)). \end{aligned}$$

Corollary 2. There is a $\% = 0^g 1$ with the property that for every sequence p, $\#\overline{U(\%p)} = \#U'(p)$.

Proof. The % in the proof of Theorem 16 has this property.

7. The Remaining Upper Bounds

In this section we use several approximations to $P(A)$, and the notion of the canonical index of a finite set A [5, pp. 69–71]. This is defined to be $\sum 2^k$ $(k \in A)$, and it establishes a one-to-one correspondence between the natural numbers and the finite sets of natural numbers. Let D_i be the finite set whose canonical index is i. We also need to use the concept of a recursive real number, which is a real x for which one can compute a convergent sequence of nested open intervals with rational end-points that contain x [5, pp. 366, 371]. This is the formal definition corresponding to the intuitive notion that a computable real number is one whose decimal expansion can be calculated. The recursive reals constitute a field.

Definition. Consider a sequence p produced by flipping an unbiased coin.

- $P(A)^t$ = the probability that (the output by time t of $U(p)$) $= A$.

  Let s be an arbitrary string.

- $P(s)^t$ = the probability that $(\forall k < |s|)$ [$k \in$ (the output by time t of $U(p)$) iff the kth bit of s is a 1].

- $P(s)$ = the probability that $(\forall k < |s|)$ $[k \in U(p)$ iff the kth bit of s is a 1$]$.

Note that $P(D_i)^t$ is a rational number that can be calculated from i and t, and $P(s)^t$ is a rational number that can be calculated from s and t.

Lemma 3.

(a) If A is finite, then $P(A) = \lim_{t\to\infty} P(A)^t$.

(b) $P(s) = \lim_{t\to\infty} P(s)^t$.

(c) $P(\Lambda) = 1$.

(d) $P(s) = P(s0) + P(s1)$.

(e) Consider a set A. Let a_n be the n-bit string whose kth bit is a 1 iff $k \in A$. Then $P(a_0) = 1$, $P(a_n) \geq P(a_{n+1})$, and $\lim_{n\to\infty} P(a_n) = P(A)$.

Theorem 18.

(a) $P(D_i)$ is a real recursive in the halting problem uniformly in i.

(b) $P(s)$ is a real recursive in the halting problem uniformly in s.

This means that given i and s one can use the oracle to obtain these real numbers as the limit of a convergent sequence of nested open intervals with rational end-points.

Proof. Note that $P(D_i) > n/m$ iff there is a k such that $P(D_i)^k > n/m$ and for all $t > k$, $P(D_i)^t \geq P(D_i)^k$. One can use the oracle to check whether or not a given i, n, m and k have this property, for there is a $\% = 0^g1$ such that $U(\%0^i10^n10^m10^k1)$ does not halt iff $P(D_i)^t \geq P(D_i)^k > n/m$ for all $t > k$. Thus if $P(D_i) > n/m$ one will eventually discover this by systematically checking all possible quadruples i, n, m and k. Similarly, one can use the oracle to discover that $P(D_i) < n/m$, that $P(s) > n/m$, and that $P(s) < n/m$. This is equivalent to the assertion that $P(D_i)$ and $P(s)$ are reals recursive in the halting problem uniformly in i and s.

Theorem 19. $P'(S_i) \succeq P(D_i)$.

Proof. It follows from Theorem 18(a) that there is a $\% = 0^g1$ with the following property. Consider a real number x in the interval between 0 and 1 and the sequence p_x that is its binary expansion. Then $U'(\%p_x) = S_i$ if x is in the open interval I_i of real numbers between $\sum_{k<i} P(D_k)$ and $\sum_{k\leq i} P(D_k)$. This shows that $c = |\%|$ has the property that $P'(S_i) \geq 2^{-c}$ (the length of the interval I_i) $= 2^{-c}P(D_i)$. (See [7, pp. 14–15] for a construction that is analogous.)

Theorem 20.

(a) $\log_2 \#\{\text{consecutive } A : H(A) < n\} = n - I'(S_n) + O(1)$.

(b) $\log_2 \#\{\text{finite } A : H(A) < n\} = n - I'(S_n) + O(1)$.

Proof. From Theorems 15, 19, 8(b)′, and 7(a), we see that

$$\begin{aligned} & n - I'(S_n) + O(1) \\ \leq\ & \log_2 \#\{\text{consecutive } A : H(A) < n\} \\ \leq\ & \log_2 \#\{\text{finite } A : H(A) < n\} \\ \leq\ & \log_2 \#\{\text{singleton } A : H'(A) < n + c\} \\ \leq\ & n + c - I'(S_{n+c}) + O(1) \\ =\ & n - I'(S_n) + O(1). \end{aligned}$$

Theorem 21. $I'(\overline{A}, A) \leq H(A) + O(1)$.

Proof. Let us start by associating with each string s an interval I_s of length $P(s)$. First of all, I_Λ is the interval of reals between 0 and 1, which is okay because $P(\Lambda) = 1$. Then each I_s is partitioned into two parts: the subinterval I_{s0} of length $P(s0)$, followed by the subinterval I_{s1} of length $P(s1)$. This works because $P(s) = P(s0) + P(s1)$.

There is a $\% = 0^g1$ which makes U' behave as follows. After reading $\%$ U' expects to find the sequence p_x, the binary expansion of a real number x between 0 and 1. Initially U' sets $s = \Lambda$. U' then works in stages. At stage k ($k = 0, 1, 2, \ldots$) U' initially knows that x is in the interval I_s, and contrives to decide whether it is in the subinterval I_{s0} or in the subinterval I_{s1}. To do this U' uses the oracle to calculate the end-points of these intervals with arbitrarily high precision, by means of the technique indicated in the proof of Theorem 18(b). And of course U' also has to read p_x to know the value of x, but it only reads the

program tape when it is forced to do so in order to make a decision (this is the crux of the proof). If U' decides that x is in I_{s0} it outputs $2k$ and sets $s = s0$. If it decides that x is in I_{s1} it outputs $2k+1$ and sets $s = s1$. Then U' proceeds to the next stage.

Why does this show that $I'(\overline{A}, A) \leq H(A) + O(1)$? From part (e) of the lemma it is not difficult to see that to each r.e. set A there corresponds an open interval I_A of length $P(A)$ consisting of reals x with the property that $U'(\%p_x) = \overline{A}$ join A. Moreover U' only reads as much of p_x as is necessary; in fact, if $P(A) > 2^{-n}$ there is an x in I_A for which this is at most $n + O(1)$ bits. Hence $I'(\overline{A}, A) \leq |\%| + H(A) + O(1) = H(A) + O(1)$.

Theorem 22.

(a) $\log_2 \#\{\text{cofinite } A : H(A) < n\} = n - I'(C_n) + O(\log I'(C_n))$.

(b) $\log_2 \#\{A : H(A) < n\} = n - I'(C_n) + O(\log I'(C_n))$.

Proof. From Theorems 17, 21, 5(a)′, 5(d)′, 13(b)′ and 7(a) we see that

$$\begin{aligned} & n - I'(C_n) + O(\log I'(C_n)) \\ \leq\ & \log_2 \#\{\text{cofinite } A : H(A) < n\} \\ \leq\ & \log_2 \#\{A : H(A) < n\} \\ \leq\ & \log_2 \#\{A : I'(A) < n + c\} \\ \leq\ & n + c - I'(C_{n+c}) + O(\log I'(C_{n+c})) \\ =\ & n - I'(C_n) + O(\log I'(C_n)). \end{aligned}$$

Corollary 3. $P'(\overline{A}) \succeq P(A)$.

Proof. By Theorems 2(b)′, 5(d)′ and 21, $H'(\overline{A}) \leq I'(\overline{A}) \leq I'(\overline{A}, A) + O(1) \leq H(A) + O(1)$. Hence $P'(\overline{A}) \succeq P(A)$.

8. The Probability of the Set of Natural Numbers Less than N

In the previous sections we established the results that were announced in Section 3. The techniques that were used to do this can also be applied to a topic of a somewhat different nature, $P(C_n)$.

$P(C_n)$ sheds light on two interesting quantities: $Q_1(n)$ the probability that a set has cardinality n, and $Q_0(n)$ the probability that the complement of a set has cardinality n. We also consider a gigantic function $G(n)$, which is the greatest natural number that can be obtained in the limit from below with probability greater than 2^{-n}.

Definition.

- $Q_1(n) = \sum P(A)$ $(\#A = n)$.
- $Q_0(n) = \sum P(A)$ $(\#\overline{A} = n)$.
- $G(n) = \max k$ $(P(C_k) > 2^{-n})$.
- Let ρ be the defective probability measure on the sets of natural numbers that is defined as follows: $\rho A = \sum Q_1(n)$ $(n \in A)$.
- Let μ be an arbitrary probability measure, possibly defective, on the sets of natural numbers. μ is said to be a C-measure if there is a function $u(n,t)$ such that $u(n,t) \geq u(n,t+1)$ and $\mu C_n = \lim_{t\to\infty} u(n,t)$. Here it is required that $u(n,t)$ be a rational number that can be calculated from n and t. In other words, μ is a C-measure if μC_n can be obtained as a monotone limit from above uniformly in n.

Theorem 23.

(a) $Q_1(n) \asymp P(C_n)$.

(b) $Q_0(n) \asymp P'(C_n)$.

(c) ρ is a C-measure.

(d) If μ is a C-measure, then $\rho A \succeq \mu A$.

(e) If $H'(S_k) < n + O(1)$, then $k < G(n)$.

(f) $H'(S_{G(n)}) = n + O(1)$.

Proof.

(a) Note that $Q_1(n) \geq P(C_n)$. Also, there is a $\% = 0^g 1$ such that $U(\%p) = C_{\#U(p)}$ for all sequences p. Hence $P(C_n) \succeq Q_1(n)$.

(b) Keep part (a) in mind. By Corollary 2, $Q_0(n) \succeq Q_1'(n) \succeq P'(C_n)$. And since $P'(\overline{A}) \succeq P(A)$ (Corollary 3), $Q_0(n) \preceq Q_1'(n) \preceq P'(C_n)$.

(c) Lemma 1(c) states that the function $Q(n)^t$ defined in Section 5 plays the role of $u(n,t)$.

(d) A construction similar to the proof of Theorem 9 shows that there is a $\% = 0^g1$ with the following property. Consider a real number x between 0 and 1 and the sequence p_x that is its binary expansion. $U(\%p_x) = C_n$ if x is in the open interval I_n of reals between μC_n and μC_{n+1}.

This proves part (d) because the length of the interval I_n is precisely μS_n, and hence $\rho S_n = Q_1(n) \geq 2^{-|\%|}\mu S_n$.

(e) By Theorem 2(c)′, if $P'(S_k) > 2^{-n}$ then $I'(S_k) < n + O(1)$. Hence by Theorem 14, there is an $l > k$ such that $P(C_l) \succeq 2^{-n}$. Thus $k < l \leq G(n + O(1))$.

(f) Note that the canonical index of C_n is $2^k - 1$. It follows from Theorem 19 that if $P(C_k) > 2^{-n}$, then $P'(\{2^k - 1\}) \succeq 2^{-n}$. There is a $\% = 0^g1$ such that $U'(\%p) = S_k$ if $U'(p) = \{2^k - 1\}$. Hence if $P(C_k) > 2^{-n}$, then $P'(S_k) \succeq P'(\{2^k - 1\}) \succeq 2^{-n}$. In other words, if $P(C_k) > 2^{-n}$ then $H'(S_k) \leq n + O(1)$. Note that by definition $P(C_{G(n)}) > 2^{-n}$. Hence $H'(S_{G(n)}) \leq n + O(1)$. Thus in view of (e), $H'(S_{G(n)}) = n + O(1)$.

Addendum

An important advance in the line of research proposed in this paper has been achieved by Solovay [8]; with the aid of a crucial lemma of D. A. Martin he shows that

$$I(A) \leq 3H(A) + O(\log H(A)).$$

In [9] and [10] certain aspects of the questions treated in this paper are examined from a somewhat different point of view.

References

[1] K. de Leeuw, E. F. Moore, C. E. Shannon and N. Shapiro, Computability by probabilistic machines, in *Automata Studies,* C. E. Shannon and J. McCarthy (Eds.), pp. 183–212. Princeton University Press, N.J. (1956).

[2] G. J. Chaitin, Randomness and mathematical proof, *Scient. Am.* **232** (5), 47–52 (May 1975).

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

[4] C. E. Shannon and W. Weaver, *The Mathematical Theory of Communication.* University of Illinois, Urbana (1949).

[5] H. Rogers, Jr., *Theory of Recursive Functions and Effective Computability.* McGraw-Hill, N.Y. (1967).

[6] R. M. Solovay, unpublished manuscript on [3] dated May 1975.

[7] S. K. Leung-Yan-Cheong and T. M. Cover, Some inequalities between Shannon entropy and Kolmogorov, Chaitin, and extension complexities, Technical Report 16, Dept. of Statistics, Stanford University, CA (October 1975).

[8] R. M. Solovay, On random r.e. sets, *Proceedings of the Third Latin American Symposium on Mathematical Logic.* Campinas, Brazil, (July 1976), (to appear).

[9] G. J. Chaitin, Information-theoretic characterizations of recursive infinite strings, *Theor. Comput. Sci.* **2**, 45–48 (1976).

[10] G. J. Chaitin, Program size, oracles, and the jump operation, *Osaka J. Math.* (to appear).

Communicated by J. T. Schwartz

Received July 1976