MINIMAL-PROGRAM COMPLEXITY OF PSEUDO-RECURSIVE AND PSEUDO-RANDOM SEQUENCES

by

R. P. Daley

Report 71-28

May, 1971

§0. Introduction

Throughout the history of the theory of recursive functions diverse hierarchies have been proposed in order to study and classify both constructive and non-constructive objects. Recently, attempts to classify recursive functions according to their complexity of computation have exposed many important aspects of the relationship between these functions and the devices used to compute them. The objects under investigation in this work will be finite and infinite binary sequences. The infinite binary sequences, which one may regard as the characteristic functions of sets, provide a means of studying the limiting behavior of finite sequences as their length increases. Several minimal-program complexity measures have been proposed (see Kolmogorov [6,7] Chaitin [1,2] Loveland [8,9]) which in a certain sense measure the information content of finite, and as a limit, infinite binary sequences. Recursive sequences are known to have extremely low minimal-program complexity and random sequences (e.g. in the sequential test sense of Martin-Löf) high complexity. In this paper the minimal-program complexity of several formulations of pseudo-recursive sequence (a pseudo-recursive sequence is one which in some sense approximates a recursive sequence) and of pseudo-random sequence. Ideally, one would hope that the pseudo-recursive sequences would have relatively low minimal-program complexity and the pseudo-random sequences relatively high complexity. However, such is not the case for these formulations suggesting that these are not adequate notions of pseudo-recursive or pseudo-random sequence at least with regard to this complexity measure. This will be discussed further in a subsequent paper entitled "Minimal-Program Complexity of Sequences with Restricted Resources", which will deal with the minimal-program complexity of sequences when the resources used for their computation are restricted.

In section 1 we present the basic definitions for the minimal program complexity, previous results and some simple lemmas which will simplify the computations in later proofs.

In section 2 we study several definitions of pseudo-recursive sequences and determine upperbounds for them in the minimal-program complexity hierarchy. We formulate two new definitions of pseudo-recursive sequences, called near re-cursive and strongly near recursive, and give tight upperbounds for them. Also considered are the almost recursive sequences defined by Vuckovic [16], the recursively approximable sequences defined by Rose and Ullian [13], and the retraceable sequences defined by Dekker and Myhill [4].

In section 3 we present an example of a pseudo-random sequence with extremely low complexity and show that it is possible to make a distinction among some types of pseudo-random sequences within the minimal-program complexity hierarchy.

§1. Minimal Program Complexity Hierarchy

The minimal program complexity was originally proposed both by Kolmogorov [6,7] and Chaitin [1,2]o If x is an infinite binary sequence then we denote by x(n) the nth member of x and by x^n the initial segment of x of length n, i.e. x^n = x(l)...x(n). If p is a string (finite sequence) then we denote by |p| the length of p (i.e. number of symbols of p). We give now Kolmogorov'[f]s original definition.

K^(x^n) = |J^.3p(|p| = *I* and G (p) = x^n) , where G
is an algorithm (computing device) and p
is a binary string (encoding of some program).
= 00, if no such p exists.

One may regard C as a digital computer and p a computer program such that when p is run on G the result is x^{11}, i.e. p contains the necessary information and procedure for the computation of x^n on G. Thus intuitively, KU_u (x^{11}) measures the information needed to compute x^n. Kolomogorov also introduced the notion of <u>conditional complexity</u>3 which measures the information (other than n) needed to compute x^n.

^ (x ^ n) = |i*.3p(|p| = *I* and G(p,n) = x^n),
where G is an algorithm and p
is a binary string.
= oo, if no such p exists.

For our investigation we will use a formulation of minimal-program complexity proposed by Loveland (see [8,9]) called the <u>uniform minimal-program complexity</u> and which is intended to insure that the only information provided by n to the program which computes x^n is that n is the length of x^n.

$$K_G(x^n;n) = \min\{|p| : |p| = \ell \text{ and } \forall i \le n.\ G(p,i) = x^i\},$$

where G is an algorithm, p is a binary string and x^i is the first i bits of x^n.

$$= \infty, \text{ if no such } p \text{ exists.}$$

One can show by the same method that Kolmogorov used for his formulation of minimal program complexity that there is a "universal" algorithm G such that for any other algorithm G' there is a constant c such that $\forall x \forall n.\ K_G(x^n;n) \le K_{G'}(x^n;n) + c_{G'}$. Therefore the minimal-program complexity of a sequence relative to two universal algorithms cannot differ by more than a constant. We fix a universal algorithm G for the remainder of this investigation and in so doing will delete the subscript. Briefly, $K(x^n;n)$ is the length of a shortest program which computes x^i, given i, for each $i \le n$.

For each x^n by considering the program which has x^n stored in its finite control and which prints out x^i, given i, one easily shows that every sequence x has a well defined minimal-program complexity for each of its initial segments.

We can associate in a natural way with each infinite binary sequence x a set of positive natural numbers X by the condition n€X4=^x(n) = 1. We say that a sequence x satisfies a property P of sets if and only if the set X associated with x satisfies P. For example, a sequence x is recursive (recursively enumerable, etc.) if and only if the set X is recursive (recursively enumerable, etc.).

By "$\overset{OO}{3}$n." and "$\overset{OO}{V}$n." we mean "there exist infinitely many n€N such that" and "for all but finitely many neN" respectively. If f : N->N U {0} then we define the complexity class named by f,

$$C[f] = \{x|\overset{OO}{V}n.K(x^{n}{}_{7}n) \pounds f(n))$$

Since we will consider classes named by functions we will make use of A notation. For example, [An.n^2] is the name of the function f such that f(n) = n^2. We will denote the greatest integer ^ n by [n].

We now present some well known properties of the minimal-program complexity hierarchy.

<u>Theorem 1.1</u>; 3c$_o$ Vx.xeC[An.n+c$_o$].

<u>Theorem 1.2</u>; x is recursive if and only if 3c.X€C[An.c].

Moreover, Loveland has constructed a separating function E,

<u>Theorem 1.3</u>: x is recursive if and only if xeC[E]•

<u>Theorem 1.4</u>: If x is recursively enumerable then there is a constant c such that xeC[An.Iog2(n)+c].

Since there are less than 2^{n+1} programs of length <£ n it follows that the number of sequences x^n for which $K(x^n;n) > n - c$ is greater than $(1-2^{0"1}) \cdot 2^n$. It therefore follows that {x|3c.x^C[An.n-c]} is a set of measure 1. Martin-Löf [11] has shown that such sequences pass all constructive stochastic tests for randomness.

<u>Theorem 1.5</u>: If 3c.x^C[An.n-c] then x is random (in the sequential test sense of Martin-Löf [11]).

In particular these sequences satisfy the strong law of large numbers $(\lim_{n\to\infty} (\frac{1}{n} S_n(x)) = \frac{1}{2}$, where $S_n(x)$ = number of 1's in x^n) and the law of the iterated logarithm $(\limsup_{n\to\infty} (\frac{S_n(x) - \frac{n}{2}}{\sqrt{n \log \log n}}) = \frac{1}{\sqrt{2}})$. Loveland and Martin-Löf have shown that random sequences necessarily have extremely high complexity, Theorem 1.6; If x is random in the sequential test sense of Martin-Löf then for every non-decreasing unbounded total recursive function £, x^C[An.n-f(n)] .

Loveland and Kolmogorov have proposed as a definition of randomness that a sequence x is random if and only if 3c.x£C[An.n-c]. Schnorr [14] has shown that there cannot exist a function f which separates the random sequences from the non-random sequences,

Theorem 1.7: If f is any unbounded non-decreasing function then there is a sequence x such that x^C[An.n-f(n)] and which does not satisfy the strong law of large numbers.

The foregoing results are very pleasing inasmuch as effectively computable sequences are characterized by the fact that they require a minimal amount of information for their computation and random sequences a maximal amount.

Many of the proofs of subsequent theorems will involve showing that the initial segment of some sequence x is computable from certain "pieces[11] of information. In order to calculate $K(x^n;n)$ these several pieces of information must be encoded into a single binary string. The following lemmas are concerned with calculating the length of this binary string in terms of the lengths of the original information strings. We make this precise in the following manner. Let N denote the set of positive natural numbers, X denote the set of all binary strings and let I : N X N -^ X and s : N -• N. We say that the infinite binary string x is uniformly computable from I in s pieces if and only if there is an algorithm ft such that for every n, Vi<;nJ(I(n_j l)^I(n,2)^,..^l(n,s(n)),i) = x^1, where * is the concatenation operation and the symbol $ (intended as a separating symbol) belongs to the alphabet of the algorithm ft. We will also say in this case that x^n **is** uniformly computable from I(n,1), .•., I(n^s(n)).

Lemma 1.8; If x is uniformly computable from I in one piece (i.e. s(n) = 1 for each n) then there is a constant c such that Vn.K(x^n7n) <; |l(n,l)| + c.

Proof; For some algorithm ft, Vn.K(B(x^n;n) <£ |l(n, 1)| and so the lemma follows by the universality of C, 3c VxVn.Kj. (x^n;n) <; K_B(x^n;n) + c.

Lemma 1.9; If x is uniformly computable from I in s pieces then there is a constant c such that

Vn.K(x^n;n) £ 2- $\sum_{i=l}^{s(n)-1}$ (|l(n,i)|+l) + |l(n,s(n))| +c.

Proof: Let (B be an algorithm such that for every n, Vi£n.B(I(n,l)*O*...*...*I(n,s(n)),i) = x^1.

Define 1 = 11^ 0 = 00 and for an arbitrary binary string IT = ^•••o^n^ where σ^1 = 0 or σ^1 = 1,

$\overline{\Pi}^{rs*} = \overline{G_1}^{r**}*...*\tilde{O}_n$ Define the information function

$I_1(n,1) = \widetilde{I(n,1)}*01*\widetilde{I(n,2)}*01*...*\widetilde{I(n,s(n)-1)}*10*I(n,s(n))$.

Clearly^ there exists an algorithm R_1 such that for every n, Vi<£n.R,$_\perp$(I$_1$(n^ 1) > i) = x^1. The lemma now follows from Lemma 1.8 and the fact that

$$|I_1(n,1)| = 2\cdot\sum_{i=1}^{s(n)-1}(|I(n,i)|+1) + |I(n,s(n))|.$$

<u>Lemma 1,10</u>: 3cVxVn.K(x^n;n) ^K(x^n|n) + 2^1og(n) + c.

<u>Proof</u>: Let I be such that I(n$_3$l) = n and | **I(n,2)** | = K(x^n|n) and G(I(n,2),n) = x^{11}. Clearly x^n is uniformly computable from I(n,l) and I(n^2).

§2. Pseudo-Recursive Sequences

Theorem 1,3 and Theorem 1.5 in essence describe the sequences at the extreme low and high ends of the minimal-program complexity hierarchy. However, only Theorem 1.4 gives any indication of the types of sequences in the middle region of the hierarchy. In this section an attempt is made to formulate a definition of pseudo-recursive sequence and to characterize such sequences in terms of the hierarchy. In the process we will encounter sequences whose complexity falls into the intermediate regions of the hierarchy.

If x and y are sequences then the sequence $x \equiv y$ is defined by the condition, $(x \equiv y)(n) = 1 \leftrightarrow x(n) = y(n)$; $\bar{x}$ by $\bar{x}(n) = 1 - x(n)$. If x is a binary sequence then we define $S_n(x) = \sum_{i=1}^{n} x(i)$, the number of 1's occurring in x^n.

The *limiting relative frequency* of a sequence x is defined by $\phi(x) = \lim_{n \to \infty} \frac{1}{n} S_n(x)$. If x and y are binary strings then we write $x \prec y$ for $\forall i \le |x| \, (x(i) = y(i))$, i.e. y is an extension of x. Also if y denotes a string then by "μy." we mean "the least string y with respect to the lexicographical ordering of binary strings such that"[11]. By "$\# j$'s." we will mean "the number of integers j such that".

One criterion for a sequence to be pseudo-recursive is that it must eventually resemble some recursive sequence. We make the following definition which was originally suggested by Loveland.

<u>Definition 2.1</u>: We say that a sequence x is <u>near recursive (n.r.)</u> if and only if there exists a recursive sequence r such that $\Phi(x=r) = 1$.

Near recursive sequences have the nice closure property that if x is near recursive and y is such that $\Phi(x=y) = 1$ then y is near recursive.

<u>Proposition 2.1</u>: If x is a sequence for which $\Phi(x) = 0$ then for every $\epsilon > 0$, $x \in C[\lambda n.\epsilon \cdot n]$.

<u>Proof</u>: For any sequence x, x^n can be computed by specifying its position (with respect to the lexicographical ordering) among all sequences of length n with exactly $s_n(x)$ 1's. It then follows by Lemma 1.9 that

$$\overset{\infty}{\forall} n.K(x^n;n) \le \log\binom{n}{s_n(x)} + 2\cdot\log(s_n(x)) + 2\cdot\log(n) + c,$$

for some constant c.

Suppose $\Phi(x) = 0$ and let $\epsilon > 0$. Choose m such that $(m+2)\cdot 2^{-m} < \epsilon$. Since $\Phi(x) = 0$,

$$\overset{\infty}{\forall} n.s_n(x) \le 2^{-m}\cdot n \text{ and also } \overset{\infty}{\forall} n.\log\binom{n}{s_n(x)} \le (m+1)\cdot 2^{-m}\cdot n.$$

Thus, $\overset{\infty}{\forall} n.K(x^n;n) \le (m+2)\cdot 2^{-m}\cdot n \le \epsilon\cdot n$.

<u>Theorem 2.2</u>: If x is near recursive then for every $\epsilon > 0$ $x \in C[\lambda n.\epsilon\cdot n]$.

<u>Proof</u>: Since x is n.r. there is a recursive r such that $\Phi(x=r) = 1$ and consequently $\Phi(x \neq r) = 0$.

Clearly, x is uniformly computable from $\bar{r}$ and $x = r$ so we have $\overset{\infty}{\forall} n.K(x^{n};n) \le K((x\oplus\bar{r})^{n};n) + 2\cdot K(\bar{r}^{n};n) + c'$. By Proposition 2.1 and Theorem 1.2 it follows that for every $\varepsilon > 0$ $\overset{\infty}{\forall} n.K(x^{n};n) \le \varepsilon\cdot n$, i.e. for every $\varepsilon > 0$ $x \in C[\lambda n.\varepsilon\cdot n]$.

Theorem 2.2 provides an upperbound for the class of near recursive sequences in the minimal-program complexity hierarchy. Since in our definition of near recursive sequence we did not specify how fast a near recursive sequence must approach some recursive sequence we are able to obtain the following result showing that the upperbound of Theorem 2.2 is a tight upperbound. We first define the set of functions

$\mathcal{F} = \{f \mid f$ is unbounded, non-decreasing, total recursive function$\}$

which represents the set of effective names for the complexity classes.

<u>Theorem 2.3</u>; If $f \in \mathcal{F}$ and $\lim\limits_{n\to\infty} \frac{f(n)}{n} = 0$ then there exists a near recursive sequence x such that $x \notin C[f]$.

<u>Proof</u>: Let y be a sequence such that $y \notin C[\lambda n.n-c]$ for some constant c. By Theorem 1.5 y is random and so $\Phi(y) = \frac{1}{2}$. We will construct the desired sequence x from y by adding sufficiently many 1's to y so that $\Phi(x) = 1$, but at a rate slow enough to insure that the difference between the complexity of x and the complexity of y will be small.

Let $f \in \mathcal{E}$. Define g by $g(n) = [2^{-1}f(m)]$ where $m = \mu p.n \leq 2 \cdot f(p)$. Clearly $g \in \mathcal{E}$ and $g(2 \cdot f(m)) \leq \frac{m}{f(m)}$. We define the sequence x as follows: We replace the nth 1 occurring in the sequence y by $g(n)$ 1's and each 0 by one 0. Since g is unbounded, $\Phi(x) = 1$ and so x is near recursive ($\Phi(x=r) = 1$, where r is the recursive sequence of all 1's).

y^n is uniformly computable from $x^{n \cdot g(n)}$ so that $\forall n.K(y^n;n) \leq K(x^{n \cdot g(n)};n \cdot g(n)) + C$, since $n \cdot g(n)$ is computable from n. Since $\overset{\infty}{\exists} n.K(y^n;n) > n - c$, $\overset{\infty}{\exists} n.2 \cdot f(n) - c - c' < K(y^{2 \cdot f(n)};2 \cdot f(n)) - c' \leq K(x^n;n)$.

We remark that the class of f's satisfying the hypothesis of Theorem 2.3 contain all the effective bounds which grow strictly slower than every constant multiple of n. Thus there exist near recursive sequences whose complexity approaches the upperbound of Theorem 2.2 as closely as can be effectively measured. The following corollary to Theorem 2.3 makes this point clearer.

Corollary 2.4: There is a near recursive sequence x such that for every $p < 1$, $x \notin C[\lambda n.n^p]$.

Proof: Let $f(n) = [n^{\frac{n}{n+1}}]$ and apply Theorem 2.3.

Because we have placed no restrictions on how fast a near recursive sequence approaches a recursive sequence we

have obtained near recursive sequences of rather high complexity. We therefore formulate a more restrictive definition of pseudo-recursive sequence. If x is a sequence then we define $1_X(n)$ = position of the nth 1 occurring in x and $0_X(n)$ = position of the nth 0 occurring in x. Thus 1_X enumerates the members of X in increasing order and 9_X enumerates the members of $\overline{X}$ in increasing order. A sequence x is <u>dense</u> if and only if for every fe&,
oo
$Vn.9_X(n)$ ^ f(n). (See Martin [10]).

<u>Definition 2.2</u>; A sequence x is <u>strongly near recursive</u> <u>(s.n.r.)</u> if and only if there is a recursive sequence r such that x s r is a dense sequence.

<u>Proposition 2.5</u>; Every strongly near recursive sequence is near recursive.

<u>Proof</u>; Let x be s.n.r., then there is a recursive r such that CD Vn.9--_--(n) ≤^ f(n), for every fe£.
Let f(n) = 2^n, then S_n(x=r) 2^n ~ log(n) - c for some constant c. Thus <£(x=r) = 1 so x is n.r.

Strongly near recursive sequences have the closure property that if x is s.n.r. and y is such that x = y is dense then y is s.n.r.

Briefly, a sequence is strongly near recursive if and only if it approaches some recursive sequence faster than can be measured by any recursive function. Because of this it is possible to obtain a lower upperbound for the complexity of strongly near recursive sequences than was obtainable for near recursive sequences.

Proposition 2.6: If x is a dense sequence then for every $f \in \mathcal{L}$, $x \in C[\lambda n.f(n)\cdot\log(n)]$.

Proof: We remark first that if x is dense then for every $f \in \mathcal{L}$, $\#j\text{'s}(j \leq n \text{ and } x(j) = 0) \leq f(n)$ for all but finitely many n. (This can be proved by considering the "inverse" g of f defined by $g(n) = \mu j.f(j) > n$.)

Let x be dense and let $f \in \mathcal{L}$, then by the above remark, $\overset{\infty}{\forall} n.(\#j\text{'s}(j \leq n \text{ and } x(j) = 0) \leq \frac{f(n)}{2})$. Thus we can compute (uniformly) x^n by specifying each $j \leq n$ for which $x(j) = 0$. It then follows by Lemma 1.9 that $\overset{\infty}{\forall} n.K(x^n;n) \leq f(n)\cdot\log(n)$.

Theorem 2.7: If x is strongly near recursive then for every $f \in \mathcal{L}$, $x \in C[\lambda n.f(n)\cdot\log(n)]$.

The proof is similar to the proof of Theorem 2.2 and so will be omitted.

If we knew that for each dense sequence x that not only $\forall f \in \mathcal{L}.\overset{\infty}{\forall} n(\Theta_x(n) \geq f(n))$ but also that there is a constant M such that for every $f \in \mathcal{L}$,

oo
Vm(#j's(f(m) £ 9 (j) £ f(m+l)) £ M) (in other words the O's of x cannot cluster together in arbitrarily large groups), then it seems reasonable that we could show that for some constant *c*, *xeC[*An.c*log(n)]. (e.g. if f(n) = 2^n then the information needed to compute x^n in this case produces the series,, log(n) + log log(n) + log log log(n) +...)• Howeverj as the proof of the following proposition shows, the 0^Ts of a dense sequence may indeed cluster together in arbitrarily large groups.

Proposition 2.8; There exists a dense sequence x such that for every constant c > 0, x^C[An.c-log(n)].

Proof; Let y be a dense sequence. We will construct a dense sequence x by regrouping the 0^Ts of y. The particular regrouping we use will enable us to show that for each constant c > 0 and for infinitely many n, x^n is different from every sequence of length n computable by a program of length <^ olog(n).

If y is a dense sequence then it can be shown that there exists a sequence {p.} such that

Pj > Pj_l + J and 9y(Pj) ~ ^Pj-1) > 2#0y(Pj) D+ 1+ 1 •

x is constructed by induction as follows; For $n \le L\, \theta_y(p_1)$ we define x(n) = y(n) .

Suppose we have constructed x^n for n <^ 9 y(p j-l). There are at most 2•2^"log(0y(pj JJ = 2-9 y(p jP programs of length ≤^ j»log(9y (pj))« On the other hand there are $\prod_{\nu=p}^{j}(\theta_y(p_j) - \theta_y(p_j-1)-k)$ strings

of length $\theta_y(p_j)$ which extend $x^{\theta_y(p_j-1)}$ and

which have exactly $j + 1$ 0's occurring between

$\theta_y(p_{j-1})$ and $\theta_y(p_j)$, all of which occur between

$\theta_y(p_j-1)$ and $\theta_y(p_j)$. Now,

$$\prod (\theta_y(p_j) - \theta_y(p_j-1)-k) \geq (\theta_y(p_j)-\theta_y(p_j-1)-j)^{j+1}$$

and by our definition of $\{p_j\}$,

$(\theta_y(p_j) - \theta_y(p_j-1)-j)^{j+1} \geq 2.\theta_y(p_j)^j$ so that

there is at least one string of length $\theta_y(p_j)$ which

extends $x^{\theta_y(p_j-1)}$ and which has exactly $j + 1$ 0's

occurring between $\theta_y(p_j-1)$ and $\theta_y(p_j)$ and which is

not computable by any program of length $\leq j\cdot\log(\theta_y(p_j))$.

We define $x^{\theta_y(p_j)}$ to be the least such sequence (with

respect to the lexicographical ordering).

It follows from our construction that for

every $k \geq j$, $x^{\theta_y(p_k)}$ is different from every program

of length $\leq j\cdot\log(\theta_y(p_k))$. Hence, for each

constant $c > 0$, $x \notin C[\lambda n.c\cdot\log(n)]$.

It can be shown by a straightforward induction

that $\forall n.\theta_y(n) \leq \theta_x(n)$ so that x is dense,

<u>Theorem 2.9</u>: There exists a strongly near recursive

sequence x such that for every constant c,

$x \notin C[\lambda n.c\cdot\log(n)]$.

<u>Proof</u>; This follows immediately from Proposition 2.8

since every dense sequence is strongly near recursive.

((x≡r) = x for the recursive sequence r of all l's).

Theorem 2.9 shows that the upperbound for strongly near recursive sequences of Theorem 2.7 is a tight one, that in fact there are such sequences whose complexity approaches that upperbound as closely as can be effectively measured. We will now consider another restriction to the definition of near recursive sequences. The notion of a recursively approximable function was formulated by Rose and Ullian [13]. If x is a sequence and g : N—>N then we define the sequence xog by (xog) (n) = x(g(n)).

Definition 2.3; A sequence x is recursively approximable if and only if for every 1-1 total recursive function g there exists a recursive sequence r such that $(xog = rog) = 1.

If we take g to be the function g(n) = n we have immediately^

Proposition 2.10; Every recursively approximable sequence is near recursive.

The next theorem shows that recursively approximable sequences extend at least as high into the complexity hierarchy as do the strongly near recursive sequences. A set X is cohesive if and only if 1) X is infinite and 2) for every recursively enumerable set Y either X 0 Y is finite or X fl¯Y is finite. A set X is quasi-cohesive if and only if X is the union of a finite (non-zero) number of cohesive sets. In [13] Rose and Ullian showed in essence that every quasi-cohesive sequence is recursively approximable.

Proposition 2,11; For every constant c there is a quasi-cohesive sequence x such that x^C[An.c«log(n)].

Proof: This proof is similar in many respects to that of Proposition 2.8. The proof relies strongly on the following fact about cohesive sets.

Fact; (Dekker and Myhill (See Rogers [12])). Every infinite set possesses a cohesive subset.

Let c > 0 and let y be a dense sequence. We define the sequence $\{p_j\}$ as follows;

$p_1 = 1$

p_{j+1} = W?(P>Pj+c+1 and $©_y(p)-©_y(p-l) > 2-G_y(p)°$ C+1 + c).

We define a sequence z as follows;

For n ≤^ $9_y(p_1)$ we define z(n) = y(n). Assuming that we have defined $z^{9Y(p_j)}$ we define $z^{Gy(P_{^+1})}$ to be the least string of length $9_y(p_{j+1})$ (with respect to the lexicographical ordering) which extends $z^{©_y(p_j)}$ and which has exactly c + 1 0's occurring between $©_y(P_{j+1} - 1)$ and $©_y(P_{j+1})$ and which is not computable by any program of length c-log(9$_y$(p$_{j+1}$))•

We are guaranteed the existence of such a string by the fact that there are less than $2^{*}G_y(P_{j+1})$ programs of length $\leq c \cdot \log(\theta_y(P_{j+1}))$ and that there are $\prod_{k=0}^{c}(\theta_y(p_{j+1}) - \theta_y(P_{j+1} - 1) - k)$ strings extending $z^{\theta_y(P_j)}$ with exactly $c + 1$ 0's occurring between $\theta_y(P_{j+1} - 1)$ and $\theta_y(P_{j+1})$.

We define the function $t(i,j)$ for each $1 \leq i \leq c + 1$ and $j \in N$ by, $t(1,j) = \mu n(\theta_y(p_j - 1) \leq n \wedge \theta_y(P_j)$ and $z(n) = 0)$.

$t(i+1, j) = \mu n(t(i, j) < n \leq \theta_y(P_j)$ and $z(n) = 0)$

Define $T_1 = \{t(1, j) \mid j \in N\}$. T_1 is infinite so by the above stated Fact there is a cohesive subset of T_1, $\hat{T}_1 = \{t(1,j) \mid j \in N_1 \subseteq N\}$.

Define $T_2 = \{t(2,j) \mid j \in N_1\}$. Similarly there is a cohesive subset of T_2, $\hat{T}_2 = \{t(2,j) \mid j \in N_2 \subseteq N_1\}$,

We thus obtain $c + 1$ cohesive sets $\hat{T}_1, \ldots, \hat{T}_{c+1}$.

Define $T_i = \{t(i, j) \mid j \in N_{c+1}\}$. T_i is cohesive since $N_{c+1} \subseteq N_i$ for $i \leq c + 1$ and every infinite subset of a cohesive set is cohesive.

Define $X = \bigcup_{i \leq c+1} T_i$. X, being the union of finitely many cohesive sets, is quasi-cohesive. Let $\bar{x}$ be the characteristic sequence of X. If $j \in N_{c+1}$, then $\bar{x}(n) = z(n)$ for $\theta_y(p_j - 1) \leq n \leq \theta_y(P_j)$ so that for infinitely many n, $K(\bar{x}^n;n) > c \cdot \log(n)$ and so $\bar{x} \notin C[\lambda n.c \cdot \log(n)]$. Therefore we have shown that for every constant c > 0 there is a quasi-cohesive sequence x such that "x^C[An.c log(n)] . But surely this also shows that for every constant c > 0 there is a quasi-cohesive sequence x such that x^C[An.c*log(n)] .

Theorem 2.12; For every constant c > 0 there is a recursively approximable sequence x such that x^C[An.c-log(n)].

Proof; This follows immediately from Proposition 2.15 since, as we remarked before, every quasi-cohesive sequence is recursively approximable•

There is a slight difference between Theorem 2.12 and Theorem 2.9 in that we are able to find a strongly near recursive sequence x such that x^C[An.c*log(n)] for any c whereas the recursively approximable sequence y for which y^C[An.olog(n)] depends on the choice of c. Theorem 2.2 provides an upperbound for the class of recursively approximable sequences in light of Propsitions 2.10. However, a tight upperbound is still unknown and it remains unclear how the additional condition in Definition 2.3 can be used to find a tight upperbound.

We now consider another definition of pseudo-recursive sequence based on the notion of almost recursive set introduced by Vuckovic [16].

<u>Definition 2.4</u>: A sequence x is <u>almost recursive</u> if and only if there is a partial recursive function φ such that if $x(n) = 1$, then $\varphi(n) = \#m$'s $(m < n$ and $x(m) = 1)$.

The following theorem gives an upperbound for the complexity of almost recursive sequences.

<u>Theorem 2.13</u>: If x is almost recursive then for every $\epsilon > 0$, $x \in C[\lambda n.(\frac{1}{2}+\epsilon)\cdot n]$.

<u>Proof</u>: Let x be almost recursive and let φ be a partial recursive function such that if $x(n) = 1$ then $\varphi(n) = \#m$'s$(m < n$ and $x(m) = 1)$.

Define $u_n = \#m$'s$(m \leq n$ and $\varphi(m)$ is defined)

$v_n = \#m$'s$(m < n$ and $x(m) = 1)$

$l_i = \#m$'s$(m \leq n$ and $\varphi(m) = i)$ for $0 \leq i \leq v_n - 1$

Clearly $\sum_{i=0} l_i \leq n$.

Given φ, u_n and v_n we can compute l_i for $0 \leq i \leq v_n - 1$. Among the l_i values m for which $\varphi(m) = i$ there is precisely one value e_i such that $x(e_i) = 1$. To specify e_i, therefore, we need $\log(l_i)$ bits of information. Since for $m \leq n$, $x(m) = 1 \Leftrightarrow m = e_i$ for some $i \leq v_n - 1$, x^n is computable from the e_i's for $i \leq v_n - 1$. Therefore since we know $|e_i|$ for each i, x^n is uniformly-computable from u_n, v_n and the concatenation of the e_i's. Thus,

$$K(x^n;n) \le 2\cdot\log(u_n) + 2\cdot\log(v_n) + \sum_{i=0}^{v_n -1} \log(l_i) + c.$$

It can be shown that $\sum_{i=0}^{v_n -1} \log(l_i) \le \frac{n}{2}$, from which it follows that for every $\epsilon > 0$, $\forall^{\infty} n. K(x^n;n) \le (\frac{1}{2} + \epsilon)\cdot n$.

The next theorem shows that this is in fact a tight upperbound.

Theorem 2.14; There exists an almost recursive sequence x such that for some constant $c > 0$, $x \in C[\lambda n. \frac{n}{2} - c]$.

Proof; Let y be a sequence such that $y \in C[\lambda n. n - c']$ for some constant c'. Define $x(2n) = y(n)$ and $x(2n+1) = 1 - y(n)$. Define $\varphi(n) = [\frac{n}{2}]$. Clearly x is almost recursive. Also y^n is uniformly computable from x^{2n} so that $K(y^n;n) \le K(x^{2n};2n) + c''$ and consequently $\exists^{\infty} n. K(x^n;n) \ge K(y^{n/2};n/2) \ge \frac{n}{2} - c$.

We consider now one further formulation of pseudo-recursive sequence due to Dekker and Myhill [4].

Definition 2.5; A sequence x is retraceable if and only if there exists a partial recursive function φ

such that if $x(n) = 1$ then 1) if $l_x(1) = n$ then $\varphi(n) = n$ and 2) if $l_x(m) = n$ for $m > 1$ then $\varphi(n) = l_x(m-1)$.

<u>Theorem 2.15</u>: If x is a retraceable sequence then there is a constant $c > 0$ such that $x \in C[\lambda n.\log(n) + c]$.

<u>Proof</u>; Let x be retraceable and let φ be a partial recursive function such that if $x(n) = 1$ then 1) if $l_x(1) = n$ then $\varphi(n) = n$ and 2) if $l_x(m) = n$ for $m > 1$ then $\varphi(n) = l_x(m-1)$. Let m_n be the largest m such that $m \le n$ and $x(m) = 1$. Given m_n we can use φ to retrace all the m's for which $m \le n$ and $x(m) = 1$. Therefore, since $m_n \le n$, by Lemma 1.8 it follows that there is a constant c such that $x \in C[\lambda n.\log(n)+c]$ ∎

We now direct our attention toward the low end of the minimal-program complexity hierarchy, in an attempt to discover the properties of sequences with extremely low complexity. However, contrary to our intuition we will find sequences with extremely low complexity which possess properties of randomness. The following theorem will play a most important part in constructing sequences of extremely low complexity.

<u>Theorem 2,16</u>; If x is a dense recursively enumerable sequence then for every fe£, xeC[f] .

<u>Proof</u>; Fundamentally the proof is very simple. Since x is r.e. there is a total recursive function h which enumerates the l's of x. Also x is dense so that for each *fe&,* there are at most f(n) 0's occurring in x^n. By specifying how many 0's occur in x^n we can determine when h has enumerated all the l's in x^n. Thus, $K(x^n|n)$ £ log(f(n)) + c £ *f*(n). However, Lemma 1.10 is of no use to us in calculating $K(x^n;n)$ since we are interested in functions fe£ with f(n) « log(n). In order to compute x^n uniformly we must know how many 0's occur in x^i for each i <^ n. We accomplish this by, having defined an inverse ge£ for f, constructing an information string 6 which will enable us to compute the number of 0's in $x^{g(m)}$ where g(m) J> n. Thus to compute x^i for each i ^ n we compute $x^{g(m_i)}$ where m_i = |j.m.g(m) J> i. We now present the formal proof.

Let fe£ and define g(n) = |im.f(m) > 3*n. Clearly ge£. Thus for some n_0, 0 (n) J> g(n) for every n ^> n_0. Also g(y?) *y* n# Let h be a total recursive function which enumerates the l's of x. We define the sequence 6 by

6(n) = l<££g(n-S$_{M}$ $_n$(6)) ^ 9 (S$_n$ $_n$(6)+l). Define t(n)

to be the largest t such that 9 .,(t) £ g(n).
Thus t(n) = #0's in $x^{g(n)}$. Furthermore it
can be shown that $s_{n+t(n)} ^ = fc^n^*$

We now show how to compute x^i for $i \leq n$.
Let m_n be the least m such that g(m) J≥ n.
We can compute $x^{\wedge}$ from g, h and $6^{m_n+t(m_n)}$
as follows:

1) Find the least k, call it $k^{\wedge}$ such that g(k) J≥ i.
Clearly k_i £ m_n.

2) Calculate h(j) for each j ^≥ 1 until the number
of values of h, which are less than or equal to $g(k_i)$,
is equal to g(k.) - S, .,, x(6). We will then know
that we have computed all the l's occurring in $x^{g(k_i)}$
and hence have computed $x^{g(k_i)}$. x^i is then simply
the first i bits of $x^{g(k_i)}$.

Thus by Lemma 1.8^ $\forall n.K(x^{\infty}_n;n)$ <^ $|6^{m_n+t(m_n)}|$+c.
Now $t(m_n)$ £ m_n + n_o and since g(£^n^ / 3) > n,
$m_n \leq$ ^£ - Hence $|6^{m_n+t(m_n)}|$ ^ 2 - ^ - + n_o and
consequently xeC[f].

Although the following proposition is a consequence of subsequent theorems, we present it here to demonstrate the usefulness of the previous theorem and to illustrate the techniques which we will be using.

Proposition 2,17; There is a sequence x such that x is not near recursive and for every $f\in\mathcal{E}$, $x\in C[f]$.

Proof; In order to construct a sequence x which is not n.r. we must insure that $\$(x=r) \neq 1$ for every recursive sequence r. Let $\{\varphi_i\}$ be an effective enumeration of all partial recursive functions. We will arrange to know which φ_i are in fact total recursive 0-1 functions since these functions yield the recursive sequences. Furthermore, we must manage our construction process so that the number of recursive functions which we are considering at any given time is sufficiently small so that the amount of information needed is extremely small.

Let y be a dense r.e. sequence and let $f\in\mathcal{E}$. By Theorem 2.16, $y\in C[\lambda n.\frac{f(n)}{3}+1]$. Also we know that there are at most $\frac{f(n)}{3}$ 0's occurring in y^n. We define the sequence δ by, $\delta(n) = 1 \Leftrightarrow \varphi_n$ is a total recursive 0-1 valued function. Define $t(i,j) = 2^{i+1} + i \cdot 2^j$ for every $i \geq 0$ and $j \geq 1$. Clearly $\forall n\, \exists i\, \exists j\, .\, t(i,j) = n$ and $t(i,j) = t(k,1)$ implies that $i = k$ and $j = 1$. We define x as follows;

$$x(n) = \begin{cases} 1 - \varphi_j(n), & \text{if } n = t(i,j) \text{ and } n > \theta_y(j) \text{ and } \delta(j)=1. \\ y(n), & \text{otherwise.} \end{cases}$$

x^n can be uniformly computed from y^n and $\delta(j)$ for each j such that $\theta_y(j) \leq n$. Therefore x^n is uniformly computable from y^n and $\delta^{f(n)/3}$ so that

Vn.K(x^n;n) ≤; J6^{f(n)/3}| + 2-K(y^n;n) + c and consequently xeC[f]•

We now show that $(x=r) ^ 1 for every recursive r. Let r be a recursive sequence so that for some j, r(n) = cp.(n). It follows that x(t(i,j)) ± cpj(t(i,j)) for every t(i,j) > 9y(j). Thus S_n(x=r) ≤^ n - 2""·).n + 9_y(j) for every n > 9_y(j). Therefore $(x=r) ≤; 1 - 2~^3 ^ 1.

§3. Pseudo-Random Sequences

In this section we examine the relationship between certain formulations of pseudo-random sequence and the minimal-program complexity hierarchy.

Interpreting each binary sequence as the sequence of outcomes of a coin tossing event, a subsequence selection rule for a sequence x is a function f which selects certain members of x in such a way that whether or not f selects the nth member of x depends only on n and the first n-1 outcomes, i.e. x^{n-1}. We make this precise. Let $\langle \cdot \rangle$ be an effective bijection between X and N.

Definition 3.1; Let $f : N \times N \to \{0,1\}$ and x be a binary sequence. We define the selection sequence y of f for x by $y(n) = f(n, \langle x^{n-1} \rangle)$. We call $x \circ l_y$ the subsequence of x selected by f.

Definition 3.2; A sequence x is Church (I) random if and only if for each infinite subsequence y of x, selected by a total recursive function, $\phi(y) = \frac{1}{2}$.

Definition 3.3; A sequence x is Church (II) random if and only if for each infinite subsequence y of x, selected by a partial recursive function, $\phi(y) = \frac{1}{2}$.

The intuitive distinction between Church (I) random and Church (II) random sequences lies in the observation that Church (I) random sequences are "random" with respect to all effective subsequence selection rules which are defined for <u>all</u> sequences, wheareas Church (II) random sequences must in addition be "random" with respect to effective subsequence selection rules which may be undefined for certain sequences.

Church (I) random sequences are the original sequences proposed by Church [3] as a definition of random sequence. Ville [15] showed that for any countable collection of selection rules one can always construct a sequence x (kollektiv) which is random with respect to these selection rules and whose initial segments always possess more 1's than 0's, so that x does not satisfy the law of the iterated logarithm. Thus there are Church random sequences ((I) and (II)) which are not "truly" random.

The following theorem, which is due to Loveland, shows that there are Church (I) random sequence with extremely low minimal-program complexity.

<u>Theorem 3.1</u>: There exists a Church (I) random sequence x such that for every $f \in \mathcal{L}$, $x \in C[f]$.

<u>Proof</u>; This proof relies strongly on the LMS algorithm, which is a well known technique for producing pseudo-random sequences by considering at each successive stage of construction successively larger finite sets of

subsequence selection rules and generating a sequence which is "random" with respect to each selection rule in the set. Let $\{\varphi_i\}$ be an enumeration of all two argument partial recursive functions. Since our selection rules are total recursive functions we can enumerate the selection rules effectively using $\{\varphi_i\}$ by specifying which φ_i are total recursive. We will increase the cardinality of the sets of selection rules at a rate slow enough to insure that the information requirements will be extremely low.

Let y be a dense r.e. sequence and let fe£. It follows that there are fewer than $\frac{f(n)}{4}$ 0's occurring in y^n and by Theorem 2.16^ yeC[An.$\frac{f(n)}{4}$]. We define the sequence 6 by^ 6(n) = 1^=^ φ_n is a total recursive 0-1 function. We construct x in stages. At each stage m we define x(n) for ne(9$_y$(m-l)^G$_y$(m)]. (Here we use (ijj] to denote [k|keN and i < k j£ j}) . Our construction process at stage m will use the set of selection rules $A_m = \{\varphi_n | n \leq Jn$ and 6(n) = 1}. It will follow that A_m is computable from 6^m and consequently x will be uniformly computable from $y^{f(n)/4}$ and $6^{f(n)/4}$, and so xeC[f].

We now give the LMS algorithm which we will use.

Define

z$_1$(n) = < jVn^-S), if 5(i) = 1; (j3j otherwise

We define the <u>patterns</u> at stage m to be the following strings *ir* of length m : *ir* = $z_1(n) \ldots z_m(n)$ where ne(9$_y$(m-l),G$_y$(m)] .

We say that the above pattern π occurs at the nth step in the construction of x. We note that only stage m patterns can occur at the steps n for $n \in (\theta_\gamma(m-1), \theta_\gamma(m)]$. We define $x(n) = 1 \iff$ the pattern occurring at the nth step has occurred at an even (or zero) number of earlier steps.

To show that x is Church (I) random let φ be a total recursive 0-1 valued function of two variables. Now $\varphi = \varphi_j$ for some j. Since for each pattern π, x takes alternating values of 0's and 1's on each succeeding occurrence of π, it follows that for every step n at every stage $m \geq j$,

$$\frac{1}{2}\sum_{i=j}^{m} 2^i - \theta_\gamma(j) \leq S_n(x\varphi_j) \leq \frac{1}{2}\sum_{i=j}^{m} 2^i + \theta_\gamma(j).$$

Therefore, since $\forall n.\theta_\gamma(n) > 2^n$, $\varphi(x\varphi_j) = \frac{1}{2}$.

Theorem 3.1 presents us with somewhat of a dilemma at this stage of our investigation. One might argue that such a result shows that there is very little relation between information and randomness, or that such sequences are very poor formulations of pseudo-randomness, or that our complexity does not accurately reflect the information content of sequences. Since it is our conviction that there is indeed a relation between information and randomness and that this complexity does accurately reflect information content, we must view this result as a rather disturbing one. However, our investigations in a subsequent paper show in essence we are able to keep our information requirements low for the computation of such sequences only by making the requirements of computation resources (time, memory, etc.) non-deterministically large.

In several of the arguments to follow we will, in addition to selecting members of a sequence x by some selection rule, also want to guess by betting (according to some betting strategy) the value of the selected member. The following proposition shows that the Church random sequences are "random" also with respect to these "betting" schemes.

<u>Proposition 3.2</u>: Let $f : N \times N \to \{0,1\}$ and $g : N^3 \to \{0,1\}$ and let x be a binary sequence. Let y be the selection sequence of f for x. Define the <u>betting sequence</u> z relative to g by

$z(n) = g(n, \langle y^{1} \wedge (n) \rangle, \langle x^{1} y(n)-1 \rangle)$. Define the functions f_1 and f_2 by

$f_1(n, \langle x^{n-1} \rangle) = 1 \iff y(n) = 1$ and $z(m) = 1$, where $m = 1_y(m)$,

$f_2(n, \langle x^{n-1} \rangle) = 1 \iff y(n) = 1$ and $z(m) = 0$, where $n = 1_y(m)$,

If $*(xo|=z) \neq \frac{1}{2}$ then $*(x|z_1) \neq \frac{1}{2}$ or $*(x|z_2) \neq \frac{1}{2}$,

where z_1 and z_2 are the subsequences of x selected by f_1 and f_2 respectively.

<u>Proof</u>; The sequences z_1 and z_2 simply select the places where we bet 1's and 0's respectively. The proposition follows from the simple observation that

if $\$(xol_{z_1}) = \tfrac{1}{2}$ and $*(xol_{z_2}) = \tfrac{1}{2}$ then $*(xol_{Y} = z) = \tfrac{1}{j}$ •

We now show that in order for the LMS algorithm construction used in Theorem 3.1 to be successful it is necessary that the sequences used in the construction be selected by total recursive functions.

<u>Theorem 3.3</u>: If x is Church (II) random then for some constant *c*, $x \not\in C[\lambda n.\log(n) - c]$.

<u>Proof</u>: Let x be a sequence such that $x \in C[\lambda n.\log(n) - 3]$. We will construct a selection sequence y and a betting sequence z such that $\$(xol_{y} = z) \geq \tfrac{1}{y_z}$. In fact we define $y(n) = 1$ for all n so that we will attempt to guess each member of x. The strategy defining z, which will rely strongly on the fact that $x \in C[\lambda n.\log(n)-3]$, is as follows.

Let $K_n^n = \{w^n | K(w^n;n) \leq \log(n) - 3\}$, then $x \in K_n$. Let w^1 be the first sequence whose computation by a program of length $\leq \log(n) - 3$ terminates. We will suppose that w_n^1 is x^n, by setting $z(j) = w(j)$, unitl we discover otherwise, i.e. until we find the first j for which $x(j) \neq w(j)$. If and when we discover that w^1_n is not x^n, we find as before the next member w_n^2 of K_n and suppose until proven otherwise that w_n^2 is x^n. We continue this procedure until the real x^n is found. Thus after at most $\tfrac{n}{j}$

incorrect guesses^ assuming $x^n \in K_n$, we are certain to find x^n. Therefore^ $S_n(x=z)$ ^>_ -j • n. We now present the formal proof.

We define z in stages. At each stage m we define z(n) for ne(e m-i'em]j where e_m = 2 by z(n) = w(n), where w is the first (with respect to time of computation) string of length e_m computable by a program of length <^ log(e_m) - 3 and which extends x11"1. Since there are at most 2- 2log^em^"3 = $\frac{e_m}{4}$ programs of length _£ log(e_m) - 3, and since x^{Gm} is computable by a program of length <^ log(e_m) - 3 there can be at most e -j^ values j, for ^G(em_i^eml for which z(j) ^ x(j). Hence

S_{e_m}(ZHX) ^|-em-f-em=f.em. Itfollcws

that $(z=x) ^>-Q- ^1^ • Clearly we can define z by z(n) = g(n^<l>^<x>) for some partial recursive function g^ since the procedure is recursive in the chosen w and w can be found by a partial recursive function. Therefore by Proposition 3.2 x is not Church (II) random.

In order to see that this result is consistent with Theorem 3.1 it must be observed that the above procedure is not total recursive. Clearly if x is any sequence such that x^C[An.log(n)-3] then for infinitely many stages m there is some ne(e m-1 n^e m] for which we are unable to find a w^{e_m} (i.e. we have exhausted K_{Q_m} and so we will search forever unsuccessfully). Thus $z(n)$ is undefined and the procedure cannot be total recursive.

Thus we are able to make a strong distinction between the class of Church (I) random sequences and the class of Church (II) random sequences by using the minimal-program complexity hierarchy. We now show that the lowerbound for the complexity of Church (II) random sequences of Theorem 3.4 is nearly a tight lowerbound.

Theorem 3.5: There is a Church (II) random sequence x such that for every $f \in £$, $x \in C[\lambda n.f(n) \cdot \log(n)]$.

Proof: The proof is very similar to that given in Theorem 3.1. Since we must be concerned with &11 partial recursive functions, to assure that the LMS algorithm proceeds successfully we must specify when a particular partial recursive function will not be defined if we attempt to use it as a selection rule. It does not suffice to specify which partial recursive functions will eventually be so undefined since by neglecting to consider them as selection rules for the values for which they are defined will in general alter the sequence which we are constructing.

We now proceed with the construction. Let y be a dense r.e. sequence and let $f \in £$. Then we have $y \in C[\lambda n.$^^-] and #0's in y^{n} £ - ^ ^ . We construct x in stages. At each stage m we construct x(n) for ne(9 y(m-1)3 9 y(m)]. For each j ≤; m, let k_j = |ak(k^9y(m) and φ_j (k,<x^{k-1}>) is undefined), where {φ_j} is an enumeration of all two-variable partial recursive functions. Let $k_j = 9_Y(m) + 1$ if no such k exists. For each j ^ m define

$$z_j(n) = \begin{cases} \varphi_j(n, \langle x^{n-1} \rangle), & \text{if } n < k_j \\ 0, & \text{otherwise.} \end{cases}$$

We say that IT = $z_1(n) \ldots z_m(n)$, for ne(9 y(m-1), 9 y(m)], is a pattern at stage m and that IT occurs at the nth step in the construction of x. We define x(n) = 1⇔ the pattern IT occurring at step n has occurred at an even (or zero) number of earlier steps.

We now show that x is Church (II) random. Let cp be a partial recursive function of two variables. Suppose that cp(n,<x^{n-1}>) is defined for every n (otherwise cp does not select an infinite subsequence of x). Now cp = φ_j for some j. Since for each pattern IT X takes alternating values of 0's and 1's on each succeeding occurrence of IT, we conclude as in Theorem 3.1 that $(xol j) = 1/2, so that x is Church (II) random.

Clearly x^n is computable from y^n and k_j for $9_y(j) \leq n$. Thus by lemma 1.9 we conclude,

$$\overset{\infty}{\forall} n. K(x^n;n) \leq K(y^n;n) + 2 \cdot (\wedge f \wedge -) \cdot \log(n) + c$$

$$\leq f(n) - \log(n)$$

Bibliography

1. Chaitin, G., "On the Length of Programs for Computing Finite Binary Sequences", JACM, JL3 (1966) No.4, pp.547-569.

2. Chaitin, G., "On the Length of Programs for Computing Finite Binary Sequences: Statistical Considerations", JACM, 16 (1969), No.l, pp.145-159.

3. Church, A., "On the Concept of a Random Sequence" Bulletin AMS, 46. (1940), pp. 130-135.

4. Dekker, J. and J. Myhill, "Retraceable Sets", Canadian J. of Math., JLO (1958), pp. 357-373.

5. Daley, R., "Pseudo-Recursiveness and Pseudo-Randomness Within Minimal Program Complexity Hierarchies", Ph.D. dissertation, Carnegie-Mellon Univ., 1971.

6. Kolmogorov, A., "Three Approaches for Defining the Concept of Information Quantity", Information Transmission, *1* (1965), pp.3-11, (also) Selected Translations in Math. Stat. and Prob., 1_3 AMS Publications (1968).

7. Kolmogorov, A., "Logical Basis for Information and Probability Theory", IEEE Transactions on Information Theory, IT-14 (1968), pp.662-664.

8. Loveland, D., "Minimal-program Complexity Measure", Conference Record ACM Symposium on Theory of Computing, May (1968) p.61-65.

9. Loveland, D., "A Variant of the Kolmogorov Concept of Complexity", Info, and Control, JL5 (1969), pp.510-526.

10. Martin, D., "Classes of R. E. Sets of Degrees of Unsolvability", Zeitschrift fur Math. Logic, JL2 (1966), pp.295-310.

11. Martin-Löf, P., "The Definition of Random Sequences", Information and Control, 9. (1966), pp.602-619.

12. Rogers, H., Theory of Recursive Functions and Effective Computability, McGraw-Hill (1967).

13. Rose, P. and J. Ullian, "Approximation of Functions on the Integers", Pacific J. of Math., JL3. (1963), pp.693-701.

14. Schnorr, C, "A Unified Approach to the Definition of Random Sequences", To Appear.

15. Ville, J., _Etude Critique de la Notion de Collectif_, Paris, Gauthiers-Villars (1939).

16. Vuckovic, V., "Almost Recursive Sets", Proceedings AMS, _23_ (1969), No. 1, pp.114-119.

Department of Mathematics
CARNEGIE-MELLON UNIVERSITY
Pittsburgh, Pennsylvania 15213

/cs
5/18/71