ITERATED LIMITING RECURSION AND THE PROGRAM MINIMIZATION PROBLEM.

L.K. Schubert

Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada.

ABSTRACT: The general problem of finding minimal programs realizing given "program descriptions" is considered, where program descriptions may specify arbitrary program properties. The problem of finding minimal programs consistent with finite or infinite input-output lists is a special case (for infinite input-output lists, this is a variant of E.M. Gold's function identification problem; another closely related problem is tne grammatical inference problem). Although most program minimization problems are not recursively solvable, they are found to be no more difficult than the problem of deciding whether any given program realizes any given description, or the problem of enumerating programs in order of nondecreasing length (whichever is harder). This result is formulated in terms of k-limiting recursive predicates and functionals, defined *by* repeated application of Gold's limit operator. A simple consequence is that the program minimization problem is limiting recursively solvable for finite input-output lists and 2-limiting recursively solvable for infinite input-output lists, with weak assumptions about the measure of program size. Gold regarded limiting function identification (more generally, "black box" identification) as a model of inductive thought. Intuitively, iterated limiting identification might be regarded as higher-order inductive inference performed collectively by an ever growing community of lower-order inductive inference machines.

KEY WORDS AND PHRASES: function identification, minimal programs, limiting recursion, inductive inference, program length measures, program properties, degree of unsolvability.

1. INTRODUCTION

A question considered by Gold [1] was for what classes of computable functions there exist machines which succeed in "identifying in the limit" any member of the class. Identifying a computable function in the limit consists of generating a sequence of "guesses" (integers) convergent to an index for the function, successive guesses being based on successively larger portions of an information sequence which lists all elements of the function. An example of a practical problem to which these concepts are relevant is the learning problem in pattern recognition. Typically an adaptive pattern recognition system is caused to "learn" a mapping from patterns to responses by presenting to it a sequence of labelled patterns, i.e., patterns with their appropriate responses. All *of* the machine's responses will conform with the desired mapping once it has identified that mapping, in the sense that it has found an algorithm (equivalently, an index) for it. Two of Gold's main results were that any r.e. class of total recursive functions is identifiable in the limit, and that the class of total recursive functions is not identifiable in the limit (hence also the class of partial recursive functions is not identifiable in the limit).

Here a modified version of Gold's problem is considered. The first modification is the replacement of information sequences by (finite or infinite) "program descriptions" which may specify arbitrary program properties. Descriptions which list input-output pairs are then regarded as a special case. The second modification is that iterated limit procedures (It-limiting recursive functionals) are admitted for program-finding, since finding suitable programs in the non-iterated limit is impossible for many classes of program descriptions. For this purpose k-limiting recursiveness is defined by straight-forward generalization of Gold's concept of limiting recursiveness. The third modification is the added requirement that programs found in the (iterated) limit be minimal according to some prescribed measure of program size. Accordingly problems of this modified type are called program minimization problems.

There are various reasons for an interest in finding minimal-length programs. In work on grammatical inference closely related to Gold's identification problem, Feldman [2] considers inference schemes which try to find "good" grammars consistent with available information about a language. One measure of goodness is the intrinsic complexity, or size, of a grammar. In terms of the function identification problem, this corresponds to finding programs which are small according to some measure of program size. Indeed, the use of small programs for inductive inference is a recurring theme in the literature (see for example Refs. 3-5); allusion is usually made to the scientific maxim knows as "Occam's Razor", according to which "it is vain to do with more what can be done with fewer" in accounting for known phenomena. The special importance of minimal programs is also suggested by the work of Kolmogorov [6], Martin-Lof [7] and others, showing that the number of symbols in the shortest program for generating a finite sequence can be taken as a measure of the information content of the sequence, and this measure provides a logical basis for information theory and probability theory.

In the following the unsolvability of most nontrivial program minimization problems is first noted. After establishment of some basic properties of k-limiting recursive predicates and functionals, it is shown that any program minimization problem is k-limiting recursively solvable if the problem of determining whether any given program satisfies any given description is k-limiting recursively solvable and programs are k-limiting r.e. in order of nondecreasing size. Simple consequences are that the problem of finding minimal programs for finite functions is limiting recursively solvable, and that the problem of finding minimal programs for arbitrary computable functions (given an explicit listing) is 2-limiting recursively solvable, with weak assumptions about the measure of program size. Lower bounds on the difficulty of these problems are already known from the work of Pager [8] and Gold [1].

Finally, the point is emphasized in the concluding remarks that limiting recursively solvable induction problems, though strictly "unsolvable" in general, are nonetheless within the reach of mechanical procedures in the important sense described by Gold, and that even problems unsolvable in the limit may be regarded as solvable in a weakened sense by an expanding community of mechanisms performing limit computations.

2. PROGRAM MINIMIZATION PROBLEMS

To fix ideas, any programmable machine M may be thought of as a 2-tape Turing machine, with one tape regarded as input-output (I/O) tape and the other as program tape. One or both tapes also serve as working tape. A computation begins with the finite-state control of the machine in a unique start state and with a program on the program tape and an input on the I/O tape. If and when the machine halts, the I/O tape expression gives the out-put. It is assumed that there is an effective 1-1 coding from tape expressions (same syntax for both tapes) onto the integers N. The program (or I/O tape expression) corresponding to code number (index) x will be written as x.

If M eventually halts with output z when supplied with program x and input y, one may write $\phi^M_x(y) = z$. If M does not halt, $\phi^M_x(y)$ is undefined. Thus M computes a partial function ϕ^M_x with program x. However, it will be convenient to think of x not merely as a program for ϕ^M_x, but as a program for any subset of ϕ^M_x. In other words, x is a program for a function ϕ provided only that $\phi^M_x(y) = \phi(y)$ for all y in the domain of ϕ; $\phi^M_x(y)$ need not be undefined for y outside that domain. If such an x exists for a given ϕ, ϕ will be said to be programmable (on M). A machine on which all partial recursive functions are programmable is universal.

A program length measure assigns a non-negative integral length to each program such that only a finite number of programs are of any particular length. A length measure need not be recursive, though this is a frequent assumption; furthermore, programs are often assumed to be effectively enumerable in order of nondecreasing length. For example, the number of elementary symbols in a program provides such a length measure,

In the following, obviously machine and length-measure dependent concepts will some-times be used without explicit reference to a particular machine or length measure. This should be kept in mind for a correct inter-pretation of the results.

A minimal program for a function ϕ is one whose length does not exeed the length any other program for ϕ. The problem of finding a minimal program (or all minimal programs) for a function ϕ given a (possibly infinite) list of the elements of ϕ, is an example of a program minimization problem. Generally, a program minimization problem is the problem of finding a minimal program (or all minimal programs) meeting the conditions listed in any "program description" belonging to some class of such descriptions. Program descriptions are loosely defined as follows. Suppose that a (possibly infinitary) logical system is given along with an interpretation based on a fixed M such that every wff in tne system expresses some program property (i.e., every wff is a unary predicate over programs). Then the wffs comprising the system will be called program descriptions. Typically a program description might specify relationships between inputs and outputs (e.g., particular input-out-put pairs), structural properties (e.g., the number of occurrences of a particular symbol in the program), operational properties (e.g., computational complexity), or combinations of such properties. If δ is a program description, a program x will be said to realize δ if x has the property expressed by δ.

more briefly, $\bar{\delta}(\bar{x})$ will be written for "$\bar{x}$ realizes $\bar{\delta}$". If an $\bar{x}$ exists such that $\bar{\delta}(\bar{x})$, then $\bar{\delta}$ will be said to be realizable. For some descriptions (such as listings of input-output pairs) the truth value of the assertion $\bar{\delta}(\bar{x})$ depends only on the function computed by M with program $\bar{x}$, i.e., $[\phi^M_x = \phi^M_y] \Rightarrow [\bar{\delta}(\bar{x}) \Leftrightarrow \bar{\delta}(\bar{y})]$. Such descriptions will be termed I/O descriptions. As examples of I/O descriptions which do not merely list functions, consider the following expressions (in a quasi - logical notation with the obvious interpretation):

(i) $[\phi^M_x(3)=9 \vee \phi^M_x(3)=11]$ & $\phi^M_x(5)$ divergent

(ii) $(\forall y)[\phi^M_x(y)$ convergent & even]

(iii) $(\forall y)[\phi^M_x(y) = \phi^M_{75}(y)] \vee (\forall y)[\phi^M_x(y+5) = \phi^M_x(y)+6]$.

Gold's identification problem can evidently be reformulated in terms of infinite descriptions (i.e., wffs belonging to an infinitary logical system) such as

(iv) $\phi^M_x(0)=0$ & $\phi^M_x(1)=1$ & $\phi^M_x(2)=4$ & $\phi^M_x(3)=9$ & ... etc.

It is assumed that descriptions can be coded numerically. If only finite descriptions are involved, an effective coding of descriptions into integers is appropriate. If infinite descriptions are involved, these can be coded as total number-theoretic functions on N by means of a 1-1 *mapping from elementary symbols into integers.* For example, consider illustration (iv) above; if the numbers from 1 to 5 are used to encode the symbols ϕ^M_x, (,), =, and &, respectively, and numerals within the description are represented by adding 6, the code sequence 1,2,6,3,4,6,5,1,2,7,3,4,7,5..., is obtained; expressed as a total function this is $\{\langle 0,1\rangle,\langle 1,2\rangle,\langle 2,6\rangle,\langle 3,3\rangle,\langle 4,4\rangle,\langle 5,6\rangle,\ldots\}$. *Thus* a mapping whose domain contains coded representations of infinite descriptions is a functional.

The coded version of a description o, whether it is finite or infinite, will be written as 6. Since no confusion can result, coded representations of descriptions will also simply be called descriptions. A set of descriptions will be called infinitely diverse if no set of programs realizina the descriptions is finite.

Theorem 1 is concerned with I/O descriptions only, while Theorems 3 and 4 will apply to arbitrary program descriptions.

Theorem 1. Let M be a universal programmable machine and let a recursive length measure be given. Then the program minimization problem is not recursively solvable for any effectively enumerable, infinitely diverse set of I/O descriptions.

Proof: Pager [8] previously noted this fact for the case when I/O descriptions specify finite functions, and remarked that the proof involves the Recursion Theorem. I formulated Theorem 1 independently and proved it, in outline, as follows. The negation of the theorem allows the construction of a program which enumerates descriptions and corresponding minimal programs until it finds a minimal program longer than itself; it then simulates that program, and contradiction results. The possibility of a program measuring its own length and then performing other arbitrary calculations follows from the existence of a recursive function g such that $\phi^M_{g(x)} = \lambda z[\phi^M_x(g(x);z)]$ for all x. This can be proved from the S-m-n Theorem and the Recursion Theorem.

To demonstrate the unsolvability of a program minimization problem (for a universal machine), it is therefore sufficient to show that the descriptions concerned include an effectively enumerable, infinitely diverse set of I/O descriptions. This implies, for example, that the program minimization problem for the singleton functions, for the finite decision functions, and even for the decision functions of cardinality 2 is unsolvable[2], whenever the length measure is recursive.

Pager [9] has shown that the last-mentioned problem is unsolvable even when the length measure is not recursive. Further, he established the surprising fact that the minimization problem is unsolvable for a certain finitely diverse set of decision functions, regardless of the length measure employed [8].

In view of Pager's results it may be asked whether the requirement that the length measure be recursive is superfluous in Theorem 1. The answer is no (although the requirement can be weakened somewhat). To prove this, it is only necessary to specify some sequence of finite decision functions such that any program is a program for at most one of these functions, plus an arbitrary procedure for obtaining a particular program for each function in the sequence; then the length measure can readily be defined to guarantee the minimality of these particular programs.

It is interesting to note that Theorem 1 still holds for certain non-1/0 descriptions. For example, suppose the minimal programs are required to operate within a certain bound on the computational complexity, apart from an arbitrary additive constant. Then it is clear that the proof of Theorem 1 is applicable without change.

These unsolvability results do not mean that all interesting program minimization problems are entirely beyond the reach of mechanical procedures, as Theorem 3 will show.

The following definitions generalize the concept of limiting recursion introduced bv Gold [1].

Definitions: Let Δ be a subset of $\Phi_1 x \ldots x \Phi_r x N^s$ where the Φ_i are sets of unary total functions and $r,s \geq 0$. A functional[3] F is k-limiting recursive on Δ if there exists a functional G recursive on $\Delta x N^k$ such that
$(\forall \delta \varepsilon \Delta)(\exists m_1)(\forall n_1 > m_1) \ldots (\exists m_k)(\forall n_k > m_k)[F(\delta) = G(\delta, n_1, \ldots, n_k)]$.
Equivalently one may write
$(\forall \delta \varepsilon \Delta)[\lim_{n_1} \ldots \lim_{n_k} G(\delta, n_1, \ldots, n_k) = F(\delta)]$.[4]
Similarly a predicate P is k-limiting decidable on Δ if there is a predicate Q decidable on $\Delta x N^k$ such that $(\forall \delta \varepsilon \Delta)(\exists m_1)(\forall n_1 > m_1) \ldots (\exists m_k)(\forall n_k > m_k)[P(\delta) \equiv Q(\delta, n_1, \ldots, n_k)]$. A set of descriptions Δ will be called uniformly k-limiting decidable (with a particular M understood) if there is a predicate P k-limitinq decidable on AxN such that for Sea, P(6,x) holds iff $\bar{\delta}(\bar{x})$. A set of integers is k-limiting r.e. if it is empty or the range of a function[5] k-limiting recursive on N. O-limiting recursive is the same as recursive, and 1-limiting recursive is abbreviated as limiting recursive.

Gold (also Putnam [10 J) has shown that

limiting decidable is equivalent to membership in the intersection of classes Σ_2 and Π_2 of the Kleene arithmetical hierarchy. It can be shown that if P is k-limiting r.e., then it is in Σ_{2k} and if P is k-limiting decidable, it is in $\Sigma_{2k} \cap \Pi_{2k}$, for all k. The converse statements are conjectured to be false for k>1; the proof (or disproof) of this conjecture is an open problem.

Lemma 1. If P and Q are predicates of at least one number variable (and possibly additional number and function variables), then $(\exists m)(\forall n>m)P(n,\ldots)$ & $(\exists m)(\forall n>m)Q(n,\ldots)$ $\equiv (\exists m)(\forall n>m)[P(n,\ldots)$ & $Q(n,\ldots)]$.

Proof. ⊃: Suppose the antecedent holds. Let $m = \max\{m_1,m_2\}$ where m_1,m_2 are numbers such that $(\forall n>m_1)P(n,\ldots)$ and $(\forall n>m_2)Q(n,\ldots)$. Then clearly $(\forall n>m)[P(n,\ldots)$ & $Q(n,\ldots)]$.
⊂: Immediate.

Corollary 1.1. If P and Q are predicates of at least k number variables (and possibly additional number and function variables), then $(\exists m_1)(\forall n_1>m_1)\ldots(\exists m_k)(\forall n_k>m_k)P(n_1,\ldots,n_k,\ldots)$ & $(\exists m_1)(\forall n_1>m_1)\ldots(\exists m_k)(\forall n_k>m_k)Q(n_1,\ldots,n_k,\ldots)$
$\equiv(\exists m_1)(\forall n_1>m_1)\ldots(\exists m_k)(\forall n_k>m_k)[P(n_1,\ldots,n_k,\ldots)$ & $Q(n_1,\ldots,n_k,\ldots)]$.

Lemma 2. Composition of k-limiting recursive functionals yields a k-limiting recursive functional.

Proof. Consider the special case $F' = \lambda\phi\psi xy[F(\phi,x,H(\psi,y))]$, where F and H are functionals k-limiting recursive on $\Phi \times N^2$ and $\Psi \times N$ respectively, and Φ, Ψ are sets of total functions. The proof is easily extended to the case where F is r-ary and s of its arguments are values of k-limiting recursive functionals $H_1,H_2,\ldots,H_s$. Since F and H are k-limiting recursive, there exist functionals G and K recursive on $\Phi \times N^{k+2}$ and $\Psi \times N^{k+1}$ respectively such that

$(\forall\phi\varepsilon\Phi)(\forall x)(\forall z)(\exists m_1)(\forall n_1>m_1)\ldots(\exists m_k)(\forall n_k>m_k)$ $[F(\phi,x,z) = G(\phi,x,z,n_1,\ldots,n_k)]$, and (1)

$(\forall\psi\varepsilon\Psi)(\forall y)(\exists m_1)(\forall n_1>m_1)\ldots(\exists m_k)(\forall n_k>m_k)[H(\psi,y) = K(\psi,y,n_1,\ldots,n_k)]$. (2)

Since $H(\Psi,y)$ is defined for all $\psi\varepsilon\Psi$ and all y, a consequence of (1) is

$(\forall\phi\varepsilon\Phi)(\forall\psi\varepsilon\Psi)(\forall x)(\forall y)(\exists m_1)(\forall n_1>m_1)\ldots(\exists m_k)(\forall n_k>m_k)[F(\phi,x,H(\psi,y)) = G(\phi,x,H(\psi,y),n_1,\ldots,n_k)]$. (3)

By Corollary 1.1, (2) & (3) then give

$(\forall\phi\varepsilon\Phi)(\forall\psi\varepsilon\Psi)(\forall x)(\forall y)(\exists m_1)(\forall n_1>m_1)\ldots(\exists m_k)(\forall n_k>m_k)[F(\phi,x,H(\psi,y)) = G(\phi,x,K(\psi,y,n_1,\ldots,n_k), n_1,\ldots,n_k)]$,

so that F' is k-limiting recursive on $\Phi \times \Psi \times N^2$. Note that it has also been shown that a recursive functional whose iterated limit is the desired composed functional can be obtained simply by composing the recursive functionals whose iterated limits are the given functionals.

Corollary 2.1. A predicate whose characteristic function(al) is expressible as a composition of k-limiting recursive function(al)s is k-limiting decidable.

For example, let the unary predicates P and Q have k-limiting recursive characteristic functions C_P and C_Q, and let $f(x,y) = xy$ for all x and y; then R defined as $R=\{x|P(x)\&Q(x)\}$ is k-limiting decidable, since $C_R(x) = f(C_P(x), C_Q(x))$ for all x and f is recursive and hence k-limiting recursive for all k.

Lemma 3. Application of the minization operator to a k-limiting decidable predicate yields a k-limiting recursive functional, provided the requisite minimal value always exists.

For example, if P is k-limiting decidable on $\Phi \times N^2$, where Φ is a class of total functions, then $\lambda\phi x[\mu y P(\phi,x,y)]$ is k-limiting recursive on $\Phi \times N$, provided $(\forall\phi\in\Phi)(\forall x)(\exists y)P(\phi,x,y)$. The notation "$\mu y\ldots$" stands for "the least y such that...".

Proof. Consider the ternary predicate P above (extension to the general case is straightforward). Let $y_{\phi x}$ be the Skolem functional in the existence criterion above, i.e., $(\forall\phi\in\Phi)(\forall x)P(\phi,x,y_{\phi x})$. Since P is k-limiting decidable, $(\forall\phi\in\Phi)(\forall x)(\forall y)(\exists m_1)(\forall n_1>m_1)\ldots(\exists m_k)(\forall n_k>m_k)[P(\phi,x,y) \equiv Q(\phi,x,y,n_1,\ldots,n_k)]$ where Q is decidable on $\Phi \times N^{k+2}$. Restrict y to $y\leq y_{\phi x}$, so that the $(\forall y)$ quantifier becomes $(\forall y\leq y_{\phi k})$. This bounded quantifier may be passed through the others: $(\forall\phi\in\Phi)(\forall x)(\exists m_1)(\forall n_1>m_1)\ldots(\exists m_k)(\forall n_k>m_k)(\forall y\leq y_{\phi x})[P(\phi,x,y) \equiv Q(\phi,x,y,n_1,\ldots,n_k)]$.

Hence

$(\forall\phi\in\Phi)(\forall x)(\exists m_1)(\forall n_1>m_1)\ldots(\exists m_k)(\forall n_k>m_k)[(\mu y\leq y_{\phi x})P(\phi,x,y)=(\mu y\leq y_{\phi x})Q(\phi,x,y,n_1,\ldots,n_k)]$.

Now since $P(\phi,x,y_{\phi x})$ holds, $(\mu y\leq y_{\phi x})P(\phi,x,y) = \mu y P(\phi,x,y)$. But $[\mu y P(\phi,x,y) = (\mu y\leq y_{\phi x})Q(\phi,x,y,n_1,\ldots,n_k)]$ is equivalent to $[\mu y P(\phi,x,y) = \mu y Q(\phi,x,y,n_1,\ldots,n_k)]$ from the definition of $y_{\phi x}$, and this is in turn equivalent to $[\mu y P(\phi,x,y)=\mu y[Q(\phi,x,y,n_1,\ldots,n_k) \vee y=n_1]]$ provided $n_1>y_{\phi x}$. Since the right side of the bracketed equality expresses a functional recursive on $\Phi \times N^{k+1}$ and since m_1 can be chosen $>y_{\phi x}$, $\mu y P(\phi,x,y)$ is k-limiting recursive on $\Phi \times N$.

Theorem 2. For any k-limiting recursive length measure, programs are (k+1)-limiting r.e. in order of nondecreasing length.

Proof. There exists a recursive function f such that $(\forall x)(\exists m_1)(\forall n_1>m_1)\ldots(\exists m_k)(\forall n_k>m_k)$
$$[f(x,n_1,\ldots,n_k) = |x|]. \tag{4}$$
Let $g(i) = (\mu x\in S_i)(\forall y\in S_i)[|x|\leq|y|]$ for all i, where $S_0=N$, $S_{i+1}=S_i-\{g(i)\}$. Thus g enumerates programs (actually, their indices) in order of nondecreasing length. Analogously let $h(i,n,n_1,\ldots,n_k) = (\mu x\in S_{in})(\forall y\in S_{in})[f(x,n_1,\ldots,n_k)\leq f(y,n_1,\ldots,n_k)]$ for all $i,n,n_1,\ldots,n_k$, where $S_{0n} = \{0,\ldots,n\}$, $S_{i+1,n} = (S_{in}-\{h(i,n,n_1,\ldots,n_k)\})\cup\{n\}$ for all i,n. As S_{in} is finite for all i,n and f is recursive, h is also recursive.

For $x\leq n$ the first quantifier in (4) is bounded and can be passed through the others:
$$(\exists m_1)(\forall n_1>m_1)\ldots(\exists m_k)(\forall n_k>m_k)(\forall x\leq n)[f(x,n_1,\ldots,n_k) = |x|]. \tag{5}$$
From the definition of h
$$(\forall i)(\forall n)(\exists m_1)(\forall n_1>m_1)\ldots(\exists m_k)(\forall n_k>m_k)[h(i,n,n_1,\ldots,n_k) = (\mu x\in S_{in})(\forall y\in S_{in})[f(x,n_1,\ldots,n_k)\leq f(y,n_1,\ldots,n_k)]]. \tag{6}$$
By Corollary 1.1, (5) & (6) give

$(\forall i)(\forall n)(\exists m_1)(\forall n_1>m_1)\ldots(\exists m_k)(\forall n_k>m_k)[h(i,n,n_1,\ldots,n_k) = (\mu x\in S_{in})(\forall y\in S_{in})[f(x,n_1,\ldots,n_k)\leq f(y,n_1,\ldots,n_k)] \;\&\; (\forall x\leq n)[f(x,n_1,\ldots,n_k) = |x|] \;\&\; (\forall y\leq n)[y,n_1,\ldots,n_k) = |y|]]$.

Since, $x,y\leq n$ in the definition of h,
$$(\forall i)(\forall n)(\exists m_1)(\forall n_1>m_1)\ldots(\exists m_k)(\forall n_k>m_k)[h(i,n,n_1,\ldots,n_k) = (\mu x\in S_{in})[|x|\leq|y|]]. \tag{7}$$
For any given i, let $m = \max\{g(j)|j\leq i\}$. Then for all $n>m$, all $j\leq i$ and all choices of $n_1,\ldots,n_k$ which guarantee the equality in (7), it is easily shown by induction on j that $S_{jn} = S_j-\{x|x>n\}$ and $h(j,n,n_1,\ldots,n_k) = g(j)$. Hence $(\forall i)(\exists m)(\forall n>m)(\exists m_1)(\forall n_1>m_1)\ldots(\exists m_k)(\forall n_k>m_k)[h(i,n,n_1,\ldots,n_k)=g(i)]$,

so that g is (k+1)-limiting recursive.

Roughly speaking, Theorem 3 states that finding minimal programs is no more difficult than enumerating programs in order of non-decreasing length or deciding whether a given program realizes a given description (whichever is harder), where the "difficulty" of a k-limiting recursive functional is k.

Theorem 3. Given: a programmable machine M, a length measure such that programs are k-limiting r.e. in order of nondecreasing length, and a set A of realizable, uniformly k-limiting decidable program descriptions.

Then the program minimization problem for A is k-limiting recursively solvable.

Proof: Since programs are k-limiting r.e. in order of nondecreasing length, there is a k-limiting recursive function f which maps N onto N such that $j>i \Rightarrow |f(j)|\geq|f(i)|$. Also, since program descriptions are uniformly k-limiting decidable, there is a predicate P such that for all $\delta\epsilon\Delta$ and $x\epsilon N$, $P(\delta,x) \Leftrightarrow \bar{\delta}(\bar{x})$.

Let $i_\delta = \mu i P(\delta,f(i))$; thus $\overline{f(i_\delta)}$ is the first minimal program realizing $\bar{\delta}$ in the sequence $\overline{f(0)},\overline{f(1)},\ldots$. By the lemmata, i_δ is k-limiting recursive on Δ.

Theorem 4. Given: a k-limiting recursive length measure such that programs are k-limiting r.e. in order of non-decreasing length and a set *A* of realizable, uniformly k-limiting decidable program descriptions.

Then the problem of finding <u>all</u> minimal programs realizing any 6eA is k-limiting recursively solvable.

Proof: Define i_δ as in the proof of Theorem 3 and let $i'_\delta = \mu i[\,|f(i)|>|f(i_\delta)|\,]$. The predicate expressed by the bracketed inequality, i.e. $\{<\delta,i>\,|\,|f(i)|>|f(i_\delta)|,\ \delta\epsilon\Delta, i\epsilon N\}$, is k-limiting decidable by Lemma 2, as f, i_δ, $||$, and the characteristic function of $\{<x,y>|x>y\}$ are all k-limiting recursive. Hence by Lemma 3 i'_δ is k-limiting recursive on Δ.

Now if the set of indices of the minimal programs realizing 6 is expressed by its canonical index

$$\iota_\delta = \sum_{i=0}^{i'_\delta} 2^{f(i)} C_p(\delta,f(i)) \ ,$$

where C_p is the characteristic function of P, with P defined as in Theorem 3, then application of Lemma 2 shows i . to be k-limiting recursive on Δ.

Note that because of the assumption in Theorem 4 that the length measure is k-limiting recursive. Theorem 3 cannot be regarded as a consequence of Theorem 4. Note also that any (k-1)-limiting recursive length measure satisfies the conditions of Theorem 4 (by Theorem 2).

Theorems 3 and 4 are the main results of this paper. The remaining theorems illustrate their application.

Theorem 5. For any recursive length measure, the problem of finding all minimal programs for finite programmable functions is limiting recursive (each finite function is assumed to be specified by a program description which lists the argument-value pairs of the function in any order).

Proof: By Theorem 2, programs are limiting r.e. in order of nondecreasing length. Let the finite function encoded by any particular $\delta\epsilon\Delta$ be $\{<y_i^\delta,z_i^\delta>|i\leq n_\delta\}$. Let $Q=\{<\delta,x,n>|$ for all $i\leq n_\delta$, M with program $\bar{x}$ and input $\bar{y}_i^\delta$ halts within n steps with output

$\bar{z}_i^\delta\}$. Clearly Q is decidable and $(\forall\delta\varepsilon\Delta)(\forall x)(\exists m)(\forall n>m)[Q(\delta,x,n)\equiv P(\delta,x)]$, where $P(\delta,x) \Leftrightarrow \bar{\delta}(\bar{x})$. Hence the problem of finding all minimal programs for any finite function is limiting recursive.

Theorem 6. Given a recursive length measure and a machine M which computes total functions only, the problem of finding minimal programs for functions programmable on M is limiting recursive (each programmable function is assumed to be specified by a program description which lists the argument-value pairs of the function in any order).

Proof: For all i, let $\langle y_i^\delta, z_i^\delta\rangle$ be the i'th argument-value pair specified by any description δ, and let $Q=\{\langle\delta,x,n\rangle \mid$ for all $i\leq n$, M with program $\bar{x}$ and input $\bar{y}_i^\delta$ computes output $\bar{z}_i^\delta\}$. Q is decidable since M always halts; if there is an m such that M fails to output $\bar{z}_m^\delta$ with program $\bar{x}$ and input $\bar{y}_m^\delta$, then $Q(\delta,x,n)$ will be false for all $n>m$; if there is no such m (so that $\bar{x}$ realizes $\bar{\delta}$), then $Q(\delta,x,n)$ is true for all n. Evidently the descriptions are uniformly limiting decidable and the theorem follows.

Gold had already shown that the problem of finding any programs {not necessarily minimal) for members of a r.e, class of total functions is limiting recursive, and Feldman [2] remarked that this can be extended in an obvious way to finding minimal programs when programs are r.e. in order of non-decreasing length. Theorem 6 strengthens this result slightly, as there are recursive length measures for which programs are not r.e. in order of nondecreasing length (e.g., define $|\ |$ so that the sequence $|0|,|1|,|2|,.$ enumerates a r.e., nonrecursive set without repetition).

Theorem 7. For any M and any limiting recursive length measure, the problem of finding minimal programs for functions programmable on M is 2-limiting recursive (as in Theorem 6, the problem is interpreted in terms of program descriptions, where any description lists the argument-value pairs of a programmable function in any order).

Proof: Let $Q=\{\langle\delta,x,n_1,n_2\rangle \mid$ for all $i\leq n_1$, M with program $\bar{x}$ and input $\bar{y}_i^\delta$ halts within n_2 steps with output $\bar{z}_i^\delta\}$, with y_i^δ, z_i^δ defined as in Theorem 6. For fixed δ,x,n_1, $Q(\delta,x,n_1,n_2)$ holds in the limit of large n_2 iff $\bar{x}$ is a program for $\{\langle y_i^\delta,z_i^\delta\rangle \mid i\leq n_1\}$. If $\bar{x}$ is a program for $\{\langle y_i^\delta,z_i^\delta\rangle \mid i\varepsilon N\}$, then $(\forall n_1)\lim_{n_2} Q(\delta,x,n_1,n_2)$; if not, there is a pair $\langle y_m^\delta,z_m^\delta\rangle$, contained in all sets $\{\langle y_i^\delta,z_i^\delta\rangle \mid i\leq n_1\}$ for which $n_1>m$, such that M with program $\bar{x}$ fails to compute output $\bar{z}_m^\delta$ for input $\bar{y}_m^\delta$; hence $(\forall n_1>m)\ \lim_{n_2} Q(\delta,x,n_1,n_2)$. It follows that the descriptions are uniformly 2-limiting decidable. Since programs are 2-limiting r.e. in order of nondecreasing length by Theorem 2, the program minimization problem is 2-limiting recursive.

Note that it is known from the work of Gold that the problem is not in general limiting recursive.

The theorem is readily generalized to descriptions which prescribe divergent computations for some inputs. The minimization problem remains 2-limiting recursive.

4. Remarks on Induction and Iterated Limiting Recursion

Deduction is concerned with the derivation of particular conseauences from oeneral premises, while induction proceeds in the opposite direction. The problem of finding an algorithm {minimal or otherwise) for a function, after inspection of some but not all values of the function, is clearly of the inductive type: the complete algorithm proposed on the basis of incomplete information expresses a generalization about the function sampled. Non-trivial inductive problems are inherently "unsolvable" in the sense that no terminating procedure exists for generating "correct" generalizations; any unverified consequence of a proposed generalization may turn out to be in error. This motivated Gold's definition of limiting recursive predicates and functionals, which are more powerful than their non-limiting counterparts. He noted that a "thinker" employing a procedure for function (or "black-box") identification in the limit and using the current guess of a function's identity as a basis for goal-directed activity would be acting on correct information eventually. In this sense, therefore, some unsolvable problems are within the reach of mechanical procedures, The most general function identification problem, however, is 2-limiting recursive. Can any mechanical system be conceived which in some sense "solves" a 2-limiting recursive problem? Not if attention is restricted to a single "thinker" generating a single sequence of guesses; however, suppose that instead of a single thinker, each of an ever growing number of such thinkers T_D,T.,... with universal computational power observes the non-terminating sequence

<Y0'Z0>,$<Y_1Z_1>$,......Which enumerates some partial recursive function O. At any time the i'th thinker T. regards as his best guess the shortest program (if any) he has been able to find which, in the time available, has given correct outputs for inputs Y_O,Y_1 an(* either no output or a correct output for any other argument tested. It is clear that each thinker will eventually be guessing a program for a subset of $; furthermore, all but a finite number of the thinkers will be guessing programs for ∲ eventually. In this iterated limiting sense the expanding community successfully identifies *. Of course there is no strategy effective in the limit for deciding in general which thinkers are guessing programs for $ at any time. To interpret third-order limit processes, one might envisage a growing number of expanding communities of the above type, each committed to a distinct value of a certain parameter. At most finitely many of the unbounded communities would in,, general be "unsuccessful". Similarly still higher-order processes could be interpreted.

Acknow1edgement

I am indebted to C.G. Morgan for several stimulating and helpful discussions and to J.R. Sampson for his many useful comments on the manuscript. I.N. Chen's help on some aspects of the paper is also gratefully acknowledged. The research was conducted under a National Research Council postdoctorate fellowship.

Footnotes

i A code into total functions would also be used for a mixture of finite and infinite descriptions; one digit, say 0, would be reserved as terminator and all function values corresponding tp points beyond the end of a finite description would be set to 0.

[2] For the finite decision functions, note that the functions computed with any finite set of programs can be diagonalized to yield a finite decision function which requires a program not in the given set. To prove infinite diversity for the 2-element decision functions, it is sufficient to show that no set of n programs can include a program for each 2-element decision function whose arguments are in a fixed set of 2 integers; but for this many arguments at least 2 of the programs must give identical results {if any), so that two unsymmetric decision functions are missed.

[3] Functions are regarded as a special case of functionals.

The equivalence follows from the fact that F(6) is independent of n.,...,n,, so that the iterated limit of G must exist.

[5] This differs from Gold's definition, which expresses limiting recursive enumerability in terms of limiting semi-decidability. However, the definitions can be shown to be equivalent (for k=l).

Actually, Feldman was concerned with "occams enumerations" of formal grammars, but the problem of finding minimal grammars for languages is essentially the same as that of finding minimal programs for decision functions.

<u>References</u>

1. E.M. Gold, "Limiting Recursion," J. Symb. Logic 30, 28-48 (1965).
2. J. Feldman, "Some Decidability Results on Grammatical Inference and Complexity," Stanford Artificial Intelligence Project Memo AI-93 (1969); also Inf. and Control 2_0, 244-262 (1972).
3. R. Solomonoff, "A Formal Theory of Inductive Inference," Inf. and Control 7, 1-22, 224-254 (1964),
4. G. Chaitin, "On the Difficulty of Computations," IEEE Trans. Inf. Theory IT-16, 5-9 (1970).
5. D.G. Willis, "Computational Complexity and Probability Constructions," J. Assoc. Comp. Mach. 17, 241-259 (1970).
6. A.N. Kolmogorov, "Three Approaches to the quantitative Definition of Information," Inform. Transmission 1, 3-11 (1965); also Int. J. Comp. Math 2, 157-168 (1968).
7. P. Martin-Lof, "The Definition of Random Sequences," Inf. and Control 9_, 602-619 (1966).
8. D. Pager, "On the Problem of Finding Minimal Programs for Tables", Inf. and Control 14, 550-554 (1969).
9. D. Pager, "Further Results on the Problem of Finding the Shortest Program for a Decision Table," presented at Symposium on Computational Complexity, Oct. 25-26, 1971; abstracted in SIGACT News, No. 13 (Dec. 1971).
10. H. Putnam, "Trial and Error Predicates and the Solution of a Problem of Mostowski," J. Symb. Logic 30, 49-57 (1965).