INDUCTIVE INFERENCE: A RECURSION THEORETIC APPROACH

by

L. Blum and M. Blum

Memorandum No. ERL-M386

13 March 1973

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

INDUCTIVE INFERENCE: A RECURSION THEORETIC APPROACH

L. Blum

Department of Mathematics
University of California, Berkeley, California 94720

M. Blum

Department of Electrical Engineering and Computer Sciences
and the Electronics Research Laboratory
University of California, Berkeley, California 94720

There are several situations that we are trying more or less to model. One arises from the standard IQ test in which a person is given a finite sequence of integers and asked to produce the next integer in the sequence. Another is provided by the following grossly simplified view of one aspect of physics: Consider a physicist who is trying to find a law to explain a growing body of experimental data. This data is presented as a set of pairs (x,y). Here, x is a description of a particular experiment, e.g., a high energy physics experiment; y is a description of the results obtained, e.g., the particles produced and their respective properties. The law describing these phenomena is essentially an algorithm for computing the function $f(x) = y$. What an inductive inference machine does is to request and obtain data in the form of pairs (x,y) and then use this to look for an algorithm i that computes (an extension of) f. Other examples arise in grammatical inference, pattern recognition, etc.

This paper develops inductive inference along the lines of Solomonoff [ ], Gold, [ ] and Feldman [ ]. What is new and distinguishes our work from theirs is our attempt to characterize the set of functions that can be identified by an inductive inference machine. In the process, we have discovered (Theorem 4, part 1) that inductive inference machines can be considerably more powerful than we previously thought possible.

The authors wish to thank Eliana Minicozzi for her vital participation in this project. This work was supported in part by a Sloan Foundation fellowship to the second author, and in part by the National Science Foundation, Grant GJ-35604X.

1. Introduction

Let N denote the natural numbers. Let $(\phi_i)_{i=0}^{\infty}$ denote an acceptable Gödel numbering of all the partial recursive functions mapping N into N [ ]. Let $(\Phi_i)_{i=0}^{\infty}$ denote a complexity measure on $(\phi_i)_{i=0}^{\infty}$ [ ]. Intuitively, $\Phi_i(x)$ is the number of steps or amount of time that algorithm i requires to compute $\phi_i(x)$.

An <u>inductive inference machine</u> is an algorithmic device or Turing machine that works as follows: First the machine is put in some initial state with its memory completely blank. From there it proceeds algorithmically except that, from time to time, the device requests an input or produces an output. Each time it requests an input, an external agency feeds the machine a pair of natural numbers (x,y) or a *, and then returns control to the machine. Each output produced by the machine is a natural number.

We next define which partial functions a given machine can infer. For this, the following terminology is useful: Let f be a partial function. Say that <u>$\vec{f}$ is an enumeration of f</u> if $\vec{f} = (a_0,a_1,\ldots)$ is an infinite sequence in which each a_i is either a pair (x,f(x)) or a *, and (x,f(x)) appears at least once in $\vec{f}$ for every $x \in$ domain (f)[1]. Let M be an inductive inference machine. We write $M[\vec{f}] \downarrow i$ and say that <u>M with input $\vec{f} = (a_0,a_1,\ldots)$ converges to i</u> if, whenever $a_0,a_1,\ldots$ are fed to M in this order, there eventually comes a time when M produces an i and then never again produces a different number. We write $M[\vec{f}] \uparrow$ and say that M with input $\vec{f}$ diverges otherwise[2]. Following Gold [ ], we say that <u>M can identify</u>

[1]One reason for the * is to have a way to represent an enumeration of the partial function f with empty domain, namely $\vec{f} = (*,*,\ldots)$.

[2]Note that there are two ways for a machine to converge and two ways for it to diverge: If $M[\vec{f}]\downarrow i$ then M with input $\vec{f}$ either produces a finite sequence of outputs, the last of which is i, or an infinite sequence of outputs that are constantly equal to i from some point on. On the other hand, if $M[\vec{f}] \uparrow$ then M with input $\vec{f}$ either produces no outputs or produces an infinite sequence of outputs with an infinite number of alterations.

$\underline{f}$ if for every enumeration $\vec{f}$ of f there is an i such that $M[\vec{f}] \downarrow i$ and ϕ_i is an extension[1] of f, i.e., $\phi_i(x) = f(x)$ for all $x \in$ domain (f). We say that M can identify a set S of partial functions if it can identify every f in S. We let S_M denote the set of all partial functions that M can identify.

Before proceeding, we give some examples.

Example 1 The set of all primitive recursive functions can be identified by an inductive inference machine. Let $\phi_{\rho(0)}, \phi_{\rho(1)}, \ldots$ be an effective enumeration of the set of all primitive recursive functions. The machine starts by requesting an input (x_0,y_0). Each time it receives a new pair (x_n,y_n), it looks for the least i such that $\phi_{\rho(i)}(x_0) = y_0, \ldots, \phi_{\rho(i)}(x_n) = y_n$, and outputs $\rho(i)$. The machine then requests the next input and continues as above. Clearly, this machine can identify all primitive recursive functions.

The simple technique used above permits one to construct a machine that can identify any recursively enumerable set $\{\phi_{\sigma(i)}\}_{i \in N}$ of partial recursive functions for which the ternary predicate $[\phi_{\sigma(i)}(x) = y]$ is decidable. For example, the set of all step-counting functions $\{\Phi_i\}_{i \in N}$ has such a predicate and therefore can be identified. The set of step-counting functions differs from the set of primitive recursive functions in that it contains arbitrarily fast-growing[2] recursive functions. However, it does not contain arbitrarily difficult 0 - 1 valued recursive functions.

[1] We do not require that $\phi_i = f$. If we did, a number of strange phenomena would arise. For example, a machine that could identify all finite partial functions would be unable to identify any partial functions with infinite domain.

[2] i.e., for every recursive h there is a recursive Φ_i such that $\Phi_i(x) > h(x)$ for all x.

Example 2. We now present a machine that is more complex than the one defined in the preceding example, and also more interesting. It can identify all quickly computable recursive functions, all recursive step-counting functions (step-counting functions defined by using Turing machines with a binary input-output code), and some arbitrarily difficult 0 - 1 valued recursive functions. This machine with input $\vec{f}$ works as follows: It first conjectures 0. Now suppose at some moment in time that its last conjecture is i. If it ever discovers that $\phi_i(y) \downarrow$ and $\phi_i(y) \not\downarrow f(y)$ for some y, then it changes its hypothesis and outputs i+1. Suppose instead it has tested and found that $\phi_i(y) \downarrow f(y)$ for all $y < x$. It next tests $\phi_i(x)$ as follows: First, it constructs an upper bound on the number of steps it will permit $\phi_i(x)$ to run. To construct this bound, it dovetails $\phi_0, \phi_1, \ldots$ looking for a j such that $\phi_j(y) \downarrow f(y)$ for all $y \leq \max\{2j,2x\}$. If and when it finds such a j, it sets $\max\{\Phi_j(y) \mid y \leq \max\{2j,2x\}\}$ to be the desired upper bound on $\Phi_i(x)$. Second it tests $\phi_i(x)$. If $\phi_i(x) \downarrow f(x)$ in the number of steps given by this upper bound, then the hypothesis i is accepted for this x. If not, then M switches from hypothesis i to hypothesis j.

The "intelligence" of this machine lies in part in its unwillingness to change hypothesis i to j unless $\phi_j(y) \downarrow f(y)$ for all $y \leq 2j$. This requirement forces j to be a reasonable hypothesis for f because program j, which is of length $\log_2 j$ (assuming a standard Gödel numbering), is not big enough to store the at least 2j bits that define f(0), ..., f(2j). Hence ϕ_j must actually compute these values of f, not just remember them.

The above machine is a special case of the machine defined in Theorem 4. As we shall see, machines like this are much more powerful than machines constructed along the general lines of example 1.

Example 3. Consider the set S of all recursive functions f with the property that the least x such that $f(x) = 1$ is itself an index for f. This is the set of so-called self-describing functions, and it is trivially identifiable. A machine can identify all $f \in S$ by simply requesting inputs and conjecturing the least x, if any, such that (x,1) has appeared as input.

Though the above machine is trivial, the set of functions it identifies is impressively large: One can show that for every recursive function g there is a self-describing recursive function f which is equal to g almost everywhere. From this it follows that the self-describing functions even include some arbitrarily difficult 0 - 1 valued recursive functions. However, no machine can identify S together with all the easily identifiable functions. In fact, let S_0 be the set of all recursive functions f such that $f(x) = 0$ for almost all x. S_0 by itself is certainly easy to identify. However, no inductive inference machine can identify $S \cup S_0$. The following proof is a diagonal argument along with an application of the recursion theorem:

Suppose M can identify S_0. Without loss of generality we suppose M initially produces an output and that it eventually makes an infinite number of input requests. Now define f by the following algorithm:

f(x) = "Determine (via the recursion theorem) the index i of this algorithm.

Let $f(x) = \begin{cases} 0 \text{ if } x < i \\ 1 \text{ if } x = i \end{cases}$. Now suppose $x > i$ and f(0), f(1), ..., f(x-1) are defined. Feed M the sequence (0,f(0), (1,f(1)), ... (x-1,f(x-1)), (x,0),(x+1,0), ...,(x+n,0), ... in this order. For each $n \geq 0$ let i_n denote the last output produced by M after seeing (0,f(0), ... (x-1,f(x-1)), (x,0), ..., (x+n,0) but before making its next input request. Dovetail the computations $\phi_{i_0}(x+1)$, $\phi_{i_1}(x+2)$, ... until an integer N is found such that $\phi_{i_N}(x+N+1) = 0$ (eventually such an N will be found since M can identify S_0 and hence can identify the almost everywhere 0 function it is being fed). Let $f(x) = f(x+1) = \ldots = f(x+N) = 0$. Let $f(x+N+1) = 1$ (so $f \neq \phi_{i_N}$)."

Clearly, f is self describing. Notice that there are infinitely many x and corresponding N such that M's last conjecture after seeing (0,f(0)), ...,(x,f(x)), ..., (x+N,f(x+N)) is i_N but $f \neq \phi_{i_N}$. Thus M cannot identify f.

One corollary of the above result is Gold's important theorem [ ] that no inductive inference machine can identify all the recursive functions. Another is a non-union theorem for inductive inference machines: $S_{M_1} \cup S_{M_2}$ is not in general identifiable.

2. Order

The purpose of this section is to show how the order in which a partial function is fed to a machine affects its final output.

Let M be an inductive inference machine and let f be a partial recursive function. We say that M can identify f by effective enumeration iff for every effective enumeration $\vec{f}$ of f there is an i such that $M[\vec{f}] \downarrow i$ and ϕ_i is an extension of f. Identification by primitive recursive enumeration[1] and by increasing enumeration[2] are defined in a similar

1. $\vec{f}$ is primitive recursive if $\vec{f} = (a_0, a_1, \ldots)$ and there is a primitive recursive function $p: N \to (N \times N) \cup \{*\}$ such that $p(n) = a_n$. Every partial recursive function has a primitive recursive enumeration.

2. $\vec{f} = ((0,f(0)), (1,f(1)), (2,f(2)), \ldots)$ is the increasing enumeration of a total function f.

way. Identification by arbitrary enumeration is synonomous with identification.

Theorem 1 (Gold) One can construct a machine M to identify by primitive recursive enumeration the set P of all partial recursive functions.

Proof:

Let $p_0, p_1, \ldots$ be an effective enumeration of all primitive recursive functions mapping N into $(N \times N) \cup \{*\}$. Clearly, each primitive recursive enumeration of $f \in P$ is of the form $(p_i(0), p_i(1), \ldots)$ for some i.

Now let ρ be a total recursive function defined by $\phi_{\rho(i)}(x)$ = "Find the least n such that $p_i(n) = (x,y)$ for some y. Output y".

If $(p_i(0), p_i(1), \ldots)$ is an enumeration of $f \in P$ then $\phi_{\rho(i)} = f$.

Now define M as follows:

```
M[f] = "Go to stage 0.
        Stage n: Request an input and let the inputs received up to now
                 be $a_0, \ldots, a_n$ in this order.
                 Find the least i such that
                 $p_i(0) = a_0, \ldots, p_i(n) = a_n$. Output $\rho(i)$ and go to stage
                 n + 1."
```

Clearly M has the desired properties. ■

This theorem has its strengths and its weaknesses. Its one remarkable strength is the fact that a machine can learn any partial recursive function if that function is fed to the machine by a primitive recursive enumeration. Another less obvious but very real strength is that a standard dovetailing of a partial recursive function yields an enumeration that is actually primitive recursive.

A weakness of this theorem is that if $\vec{f}$ is a primitive recursive enumeration of a difficult-to-compute recursive function, then a very large number of repetitions will occur in $\vec{f}$, and in some applications this is not a natural way to feed a function to a machine. In fact, a device built to learn sequences such as 314159... may be viewed as an inductive inference machine that is fed the function (0,f(0)), (1,f(1)), (2,f(2)), ... in increasing order (of the domain). It is easy to see that any set of recursive functions that can be identified by increasing enumeration can also be identified by arbitrary enumeration; hence the identification of sequences does not fall into the category of Theorem 1.

We will next show that if a set of partial recursive functions can be identified by effective enumeration then it can be identified by arbitrary enumeration. Furthermore, the machine to do this identification can even be made order independent: M is <u>order independent</u> if it has the property that for every partial function f, if $M[\vec{f}] \downarrow i$ for some enumeration $\vec{f}$ of f, then $M[\vec{f}] \downarrow i$ for every enumeration $\vec{f}$ of f.

Before proceeding we will need a few definitions: First, let $\vec{\sigma} = (a_0, a_1, \ldots, a_n)$ be a finite sequence such that each $a_i \in (N \times N) \cup \{*\}$. Let Σ denote the set of all such finite sequences, $\vec{\sigma}$. If $\vec{\sigma}_1 = (a_o,\ldots,a_n)$ and $\vec{\sigma}_2 = (b_o,\ldots,b_m)$, then let $\vec{\sigma}_1 \cdot \vec{\sigma}_2 = (a_o,\ldots,a_n,b_o,\ldots,b_m)$. Say $\vec{\sigma}$ is an <u>extension</u> of $\vec{\sigma}_1$ if for some $\vec{\sigma}_2$, $\vec{\sigma} = \vec{\sigma}_1 \cdot \vec{\sigma}_2$. Say <u>$\vec{\sigma} \in \Sigma$ is contained in (a partial function) f</u> if for each pair (x,y) in $\vec{\sigma}$, f(x) = y; $\vec{\sigma}$ is <u>consistent</u> iff it is contained in some partial function.

Now for convenience we suppose for the rest of this section that each inductive inference machine initially produces some output and makes an infinite number of input requests. If M is an inductive inference machine and if $\vec{\sigma} = (a_o, \ldots, a_n)$ is in Σ, we let $M[\vec{\sigma}]$ denote the last output produced by M after it has seen $a_o, \ldots, a_n$ in this order, but before M makes its next (n+2nd) request. By our supposition on M, $M[\vec{\sigma}]$ is always defined and computable[1].

Lemma: Let M be an inductive inference machine and let f be a partial recursive function that it can identify by effective enumeration. Then there is a $\vec{\sigma}$ contained in f such that $M[\vec{\sigma}'] = M[\vec{\sigma}]$ for all extensions $\vec{\sigma}'$ of $\vec{\sigma}$ contained in f.

Proof: Suppose the lemma does not hold for some M and for some f that it can identify by effective enumeration. We shall get a contradiction by constructing an effective enumeration $\vec{f}$ of f such that $M[\vec{f}] \uparrow$:

Let $(a_0, a_1, \ldots)$ be some fixed effective enumeration of f. Effectively construct a sequence $\vec{\sigma}_0, \vec{\sigma}_1, \ldots$ of sequences contained in f as follows:

"Go to stage 0.

Stage 0: Let $\vec{\sigma}_0 = (a_0)$. Go to stage 1.

Stage n+1: $\vec{\sigma}_n$ is contained in f. Look for an extension $\vec{\sigma}_n'$ of $\vec{\sigma}_n$ contained in f such that $M[\vec{\sigma}_n'] \neq M[\vec{\sigma}_n]$ ($\vec{\sigma}_n'$ must exist else $\vec{\sigma}_n$ is the desired $\vec{\sigma}$). Let $\vec{\sigma}_{n+1} = \vec{\sigma}_n' \cdot (a_{n+1})$

Go to stage n + 2."

For each n, $\vec{\sigma}_n$ is contained in f, a_n belongs to $\vec{\sigma}_n$, and $\vec{\sigma}_{n+1}$ extends $\vec{\sigma}_n$. Thus $\lim_{n\to\infty} \vec{\sigma}_n$ is a well defined effective enumeration of f which we

[1] Whenever the technical tool $M[\vec{\sigma}]$ is used in this paper we assume it is well defined e.g., we implicitly make the above assumptions on M. The main results are independent of these assumptions.

call $\vec{f}$. But when M is fed $\vec{f}$ it produces an infinite sequence of outputs with an infinite number of alterations (i.e. $M[\vec{f}]$ changes its mind an infinite number of times). So $M[\vec{f}]\uparrow$. ∎

<u>Remark</u>: Let M be an inductive inference machine, let f be a partial recursive function that M can identify by effective enumeration, and let $\vec{\sigma}$ be the finite sequence given by the lemma. Then $M[\vec{\sigma}]$ is an index for an extension of f. To see this , first note that there is certainly an effective enumeration $\vec{f}$ of f with initial subsequence $\vec{\sigma}$. Since M can identify f by effective enumeration, M with input $\vec{f}$ converges to an index for an extension of f. By the properties of $\vec{\sigma}$, this index is equal to $M[\vec{\sigma}]$.

<u>Theorem 2</u> Let M be an inductive inference machine and let S be the set of all partial recursive functions that M can identify by effective enumeration. Then there is a machine M' uniform in M that can identify (by arbitrary enumeration) every $f \in S$. Furthermore, M' is order independent.

<u>Proof</u>: We design a machine M' with the property that whenever it is fed an enumeration $\vec{f}$ of $f \in S$, it looks for a sequence $\vec{\sigma}$ with the properties of the lemma and then converges to $M[\vec{\sigma}]$:

First let A be a subset of $(N \times N) \cup \{*\}$. Let $\Sigma_A = \{\vec{\sigma} \in \Sigma |$ each a_i in $\vec{\sigma}$ belongs to $A\}$. Note that if $A \subseteq A'$ then $\Sigma_A \subseteq \Sigma_{A'}$. Now fix some standard effective enumeration of Σ.

$M'[\vec{f}]$ = "Go to stage 0.

<u>Stage 0</u>: Request an input, call it a_0. Let $\vec{\sigma}_0 = (a_0)$ and output $M[\vec{\sigma}_0]$. Go to stage 1.

stage n+1: Request an input and then let A_{n+1} be the set of inputs received so far. Let $\vec{\sigma}_{n+1}$ be the first sequence in the enumeration of Σ such that (i) $\vec{\sigma}_{n+1} \in \Sigma_{A_{n+1}}$ and (ii) $M[\vec{\sigma}'] = M[\vec{\sigma}_{n+1}]$ for all extensions $\vec{\sigma}'$ of $\vec{\sigma}_{n+1}$ in $\Sigma_{A_{n+1}}$ that occur in the effective enumeration of Σ either before $\vec{\sigma}_{n+1}$ or among the n+1 elements immediately after $\vec{\sigma}_{n+1}$.
If $\vec{\sigma}_{n+1} = \vec{\sigma}_n$ output nothing.
Otherwise, output some number different from $M[\vec{\sigma}_n]$ and then output $M[\vec{\sigma}_{n+1}]$. Go to stage n+2."

M' is order independent: For each partial function f, one of the following two cases must apply.

Case 1) Each $\vec{\sigma}$ contained in f has an extension $\vec{\sigma}'$ contained in f such that $M[\vec{\sigma}'] \neq M[\vec{\sigma}]$: In this case, whenever M' is fed an enumeration $\vec{f}$ of f, each $\vec{\sigma}_n$ produced must necessarily be discarded at some later stage. So M' will produce an infinite sequence of outputs with an infinite number of alterations. So $M[\vec{f}] \uparrow$.

Case 2) There is a $\vec{\sigma}$ contained in f such that for all extensions $\vec{\sigma}'$ of $\vec{\sigma}$ contained in f, $M[\vec{\sigma}'] = M[\vec{\sigma}]$. Let $\vec{\sigma}$ denote the first such sequence in the enumeration of Σ: Suppose $\vec{f}$ is an enumeration of f and M is fed $\vec{f}$. By some stage all the elements of $\vec{\sigma}$ will be fed to M and so from that stage on, the search for $\vec{\sigma}_n$ will never get past $\vec{\sigma}$. If $\vec{\sigma}_n$ occurs before $\vec{\sigma}$, then (by the choice of $\vec{\sigma}$) $\vec{\sigma}_n$ must eventually be discarded never to be chosen again. Hence $M'[\vec{f}] \downarrow M[\vec{\sigma}]$ for every enumeration $\vec{f}$ of f.

Now assume that M can identify f by effective enumeration. We show that M' can identify f: By the lemma, case 2 above applies, and so for each enumeration $\vec{f}$ of f, $M'[\vec{f}] \downarrow M[\vec{\sigma}]$. By the remark, $M[\vec{\sigma}]$ is an index for f. So M'can identify f. ∎

As a consequence of the above result, we shall assume without loss of generality that all our inductive inference machines (unless otherwise stated) are order-independent[1]. We shall also write M[f] without mentioning the order in which f is fed to M.

3. Strong identification.

Let S be a set of partial functions, and let M be an inductive inference machine. We say that M is strong on S iff for each $f \in S$, $M[f] \downarrow$ if and only if M can identify f.

A systematic study of strong machines and their naturalness for inductive inference has been made by E. Minicozzi [ ]. (Her paper also has a number of other interesting results on inductive inference.)

Suppose M is strong on S. Then whenever M with input $f \in S$ conjectures an integer i such that ϕ_i is not an extension of f, then M must eventually change its mind: A strong machine thus informs the world of its mistakes.

It is easy to see that the machine of example 1 which identifies the primitive recursive functions is strong on the set of all partial functions. As we shall see, the machine of example 2 is strong on R, the set of all total récursive functions, but not on P, the set of all partial recursive functions. In fact, it follows from Theorem 3 that

[1]These machines define partial functionals.

machines that can identify arbitrarily complex 0-1 valued recursive functions cannot be strong on P. Finally, no machine for identifying the set of self-describing functions of example 3 can even be strong on S_0, the set of almost everywhere 0 functions.

Strong machines have a very useful union property:

Union theorem (E. Minicozzi): Let M_1 and M_2 be two inductive inference machines that are strong on S. Then one can uniformly construct a new machine M that is strong on S and as powerful[1] as both M_1 and M_2 on S.

Proof: First note that given an inductive inference machine M one can easily construct a machine M' (uniform in M) that is at least as powerful as M, that initially produces an output, and that has the property that after n steps of computation, if it (M') outputs an index i_n, then $i_n \geq n$.[2]

Construct M_1' and M_2' from M_1 and M_2 as indicated above. Now M works as follows: It runs both M_1' and M_2' and makes its own current (i.e. last) output be identical to whichever current output of M_1' and M_2' is smallest.

Thus if $M[\vec{f}] \downarrow i$ then either $M_1'[\vec{f}] \downarrow i$ or $M_2'[\vec{f}] \downarrow i$, while if either $M_1'[\vec{f}]$ or $M_2'[\vec{f}]$ converges, then so does $M[\vec{f}]$. If follows (by the first implication) that M is strong on S; and (by the second implication and strongness) that if $f \in S$ can be identified by either M_1 or M_2, then it can be identified by M. ∎

1. M is (at least) as powerful as M_1 (on S) iff M can identify those partial functions (in S) that M_1 can identify.

2. M' is not necessarily order independent.

In the next two sections we characterize precisely what can be identified by machines that are strong on P and on R.

4. Strong Identification on P

We first note that if M is strong on the set of all partial functions then it is strong on the set P of all partial recursive functions and hence on the set S_{finite} of all finite partial functions. It is also easy to see that strongness on S_{finite} implies strongness on the set of all partial functions. Hence, the three notions of strongness are equivalent.

The main theorem of this section characterizes the maximum power of machines strong on P. These machines can identify, for example, step-counting functions, real time computable functions, and almost everywhere polynomial functions. However, they cannot identify complex (slowly computable) 0-1 valued recursive functions.

For the characterization we will need the following complexity theoretic notion:

Suppose h is a recursive function of two variables. We say that a partial function f is h-honest[1] if it has an extension ϕ_i such that $\Phi_i(x) \leq h(x,\phi_i(x))$ for almost all $x \in$ domain f. An h-honest function f is one that can be computed within the amount of time, modulo h, required to read the input x and print the output f(x). We let S_h denote the set of all h-honest partial functions.

[1]This definition of h-honesty is a generalization of A. Meyer's definition (see [ ]). We point out the following peculiarity of this definition: Suppose $f(x) = \begin{cases} 1 \text{ if } \phi_x(x) \downarrow \\ \uparrow \text{ otherwise} \end{cases}$. Then for each recursive h and for each algorithm i for computing f(i.e., $f=\phi_i$) there are infinitely many x such that $\Phi_i(x) > h(x,\phi_i(x))$. However, for any reasonable h, f is h-honest since f can be extended to the constant function, g(x) = 1.

Theorem 3 Let M denote an inductive inference machine that is strong on P. Let h denote a recursive function of two variables.

1. For every h there is an M uniform in h such that for all partial functions f, f is h-honest $\Rightarrow$ M can identify f, i.e., $S_h \subseteq S_M$.

2. For every M there is an h uniform in M such that for all partial functions f, M can identify f $\Rightarrow$ f is h-honest, i.e., $S_M \subseteq S_h$.

Proof

1. A recursive h is given. First note that if $f \in S_h$, then there exist integers i, m such that ϕ_i is an extension of f and $\Phi_i(x) < \max\{m,h(x,f(x))\}$ for all x in the domain of f. This suggests the following definition of M:

M[f] = "Go to stage 0.

Stage $n = \langle i,m \rangle$:[1] Output i. Check that (i) $\Phi_i(x) < \max\{m,h(x,f(x))\}$ and (ii) $\phi_i(x) = f(x)$ for all $x \in$ domain (f). If and when this check fails for some x, go to stage n+1."

We first show that M is strong on P: Let $f \in P$ and suppose $M[\vec{f}] \downarrow i$. Then M with input $\vec{f}$ must enter stage $\langle i,m \rangle$ for some m and never leave it. But in this stage, M checks that $\phi_i = f$ on the domain of f. Hence ϕ_i must be an extension of f. This shows that M is strong on P. We next show that $S_h \subseteq S_M$: Let $f \in S_h$. Let $n = \langle i,m \rangle$ be the least integer such that ϕ_i is an extension of f and $\Phi_i(x) < \max\{m,h(x,f(x))\}$ for all $x \in$ domain (f). It is easy to see that for any enumeration $\vec{f}$ of f, $M[\vec{f}]$ can never get past stage n. It follows that $M[\vec{f}]$ must converge, and since M

1. $\langle\ \rangle$ is an effective 1-1 map from N×N onto N.

is strong on P (and hence strong on the partial functions) it must converge to a correct index for an extension of f.

2. We make some preliminary remarks: An inductive inference machine M is said to have the overkill property iff for each consistent $\vec{\sigma} \in \Sigma$, if $M[\vec{\sigma}] = i$ then $\phi_i(x) = y$ for all pairs (x,y) in $\vec{\sigma}$. Clearly, if M has the overkill property, then M is strong on all partial functions.

Lemma 1 Suppose M can identify S_{finite}. Then there is an M' uniform in M that is as powerful as M and has the overkill property.

Proof: M'[f] = "Output 0 and go to stage 0.

```
Stage n: Request an input until a pair is received. Suppose
$(x_0,y_0),\ldots,(x_n,y_n)$ are the pairs received until now. Feed M
this sequence followed by a sequence of *'s until M produces an
integer i such that $\phi_i(x_0) = y_0,\ldots,\phi_i(x_n) = y_n$.
Output i and go to stage n+1."
```

¤

Lemma 2: Suppose M is strong on P. Then there is an M' uniform in M that is as powerful as M and has the overkill property.

Proof: It is easy to see that there is an inductive inference machine M_1 that is strong on the partial functions and that can identify S_{finite}. By the union theorem there is a machine M_2 uniform in M_1 and M that can identify S_{finite} and is as powerful as M. Apply lemma 1 to M_2. ¤

We are now ready to prove 2. Suppose M is strong on P and so, without loss of generality, suppose M has the overkill property. Let < be an effective ordering of N×N, and for each $(x,y) \in N\times N$ let $\Sigma(x,y)$ be the finite set of all finite consistent sequences of the form $\vec{\sigma} = ((x_0,y_0),$ $\ldots,(x_n,y_n), (x,y))$ such that $(x_0,y_0)<\ldots< (x_n,y_n) < (x,y)$. $\Sigma(x,y)$ is a finite set and by the overkill property, $\phi_{M[\vec{\sigma}]}(x) = y$ for each $\vec{\sigma} \in \Sigma(x,y)$.

So the function $h(x,y) = \max\{\Phi_{M[\vec{\sigma}]}(x) \mid \vec{\sigma} \in \Sigma(x,y)\}$ is recursive. We now show that

$S_M \subseteq S_h$: Suppose $f \in S_M$. If the domain of f is finite then $f \in S_h$ trivially. If the domain of f is infinite, let $\vec{f} = ((x_0,y_0), (x_1,y_1), \ldots)$ be the enumeration of f induced by the order $<$ on $N \times N$. $M[\vec{f}]$ converges to an index i (for an extension of f). For sufficiently large n, $M[(x_0,y_0), \ldots, (x_n,y_n))] = i$. Now by definition of h it follows that for sufficiently large n, $\Phi_i(x_n) \leq h(x_n,\phi_i(x_n))$. So $f \in S_h$. ■

Corollary: Suppose S is a set of partial functions such that $S \cup S_{\text{finite}}$ can be identified by an inductive inference machine. Then there is a recursive function h such that $S \subseteq S_h$.

5. Strong Identification on R

The machine of Theorem 3 makes up conjectures, i, and then tests each i on all inputs x = 0, 1, 2, ... A great weakness of this machine is that it uses an *a priori* upper bound $h(x,f(x))$ to determine the number of steps it permits $\phi_i(x)$ to take: If $\Phi_i(x)$ exceeds this bound, then hypothesis i is automatically rejected. The more sophisticated machine we shall define in this section uses an *a posteriori* upper bound. To develop this bound, the machine has a built-in formal criterion of what constitutes a reasonable alternative hypothesis j to a given hypothesis i, given that $\phi_i(y) \downarrow f(y)$ for all $y < x$(e.g. the criterion of section 1, example 2 is that $\phi_j(y)$ should converge to $f(y)$ for all $y \leq \max\{2x,2j\}$). The machine looks for such a j and then uses it to construct an upper bound on $\Phi_i(x)$ (e.g. the bound of section 1, example 2, is $\max\{\Phi_j(y) \mid y \leq \max\{2j,2x\}\}$. If $\Phi_i(x)$ exceeds this bound, then the machine switches from hypothesis i to hypothesis j.

We shall give a few preliminary definitions now that lead up to the formal criteria mentioned above.

First, we shall need the idea of a recursive operator [ ]: Informally, any such operator $\mathcal{O}$ is an effective mapping from partial functions g to partial functions h;it is defined in terms of an algorithmic device, $\mathcal{D}_{\mathcal{O}}$. For our purpose, we shall assume that $\mathcal{D}_{\mathcal{O}}$ is fed the sequence g(0), g(1), ... in natural order up to the least y, if any, not in the domain of g. We interpret the output integers of $\mathcal{D}_{\mathcal{O}}$, in the order of their generation, to be h(0), h(1), ... We shall be particularly concerned with those recursive operators $\mathcal{O}$ having the special property that they map R into R.

Let $\mathcal{O}$: R $\to$ R be a recursive operator. Let g be a partial function. We shall write $\mathcal{O}(g)$ for the partial function obtained by applying $\mathcal{O}$ to g, and $\mathcal{O}(g,x)$ for $\mathcal{O}(g)$ (x). For f $\in$ R, we say that <u>f is everywhere $\mathcal{O}$-compressed</u> if there is an algorithm i that computes f in such a way that, given any algorithm j for computing f, $\Phi_i(x) \leq \mathcal{O}(\Phi_j, \max\{i,j,x\})$ for all x. The above index i is called a <u>compression index</u> (with respect to $\mathcal{O}$) for f.

<u>Theorem 4</u> Let M denote an inductive inference machine that is strong on R. Let $\mathcal{O}$ denote a recursive operator that maps R $\to$ R.

1. For every $\mathcal{O}$ there exists an M uniformly in $\mathcal{O}$ such that for all f $\in$ R, f is everywhere $\mathcal{O}$-compressed $\Rightarrow$ M can identify f.

2. For every M there exists an $\mathcal{O}$ uniformly in M such that for all $f \in R$, M can identify $f \Rightarrow f$ is everywhere $\mathcal{O}$-compressed.

<u>Proof</u> 1. Let $\mathcal{O}: R \to R$ be a recursive operator. We shall construct an M that can identify every $f \in R$ that is everywhere $\mathcal{O}$-compressed. Basically, one would like to construct M so that M[f] looks for a compression index for f, and converges when it finds one. To do so, M[f] should converge if it finds an i such that $\phi_i = f$ and such that if $\phi_j = f$ then $\Phi_i(x) \le \mathcal{O}(\Phi_j, \max\{i,j,x\})$ for all x. The difficulty here is that M has no way of checking that $\phi_j = f$ in the case that $\Phi_i(x) > \mathcal{O}(\Phi_j, \max\{i,j,x\})$ for some x. A patching argument will be used to overcome this difficulty.

<u>Definition</u> Let $\mathcal{O}: R \to R$ be a recursive operator, let $g \in P$, and let $x \in N$.

Define $\Delta\mathcal{O}(g,x)$ = the least n such that $g(y)\downarrow$ for all $y < n$ and $\mathcal{O}(g,x)\downarrow$ when $\mathcal{D}_{\mathcal{O}}$ is fed the sequence $(g(y))_{y < n}$ if such n exists, ∞ otherwise.

<u>Remarks</u> (i) If $\mathcal{O}(g,x)\downarrow$ then $\Delta\mathcal{O}(g,x) < \infty$ and $\Delta\mathcal{O}(g,x)$ is the least number of (g-valued) inputs $\mathcal{D}_{\mathcal{O}}$ needs to output $\mathcal{O}(g,x)$.

(ii) If $\mathcal{O}(g,x)\uparrow$ then $\Delta\mathcal{O}(g,x) = \infty$ and (since $\mathcal{O}: R \to R$) $g(y)\uparrow$ for some y.

(iii) Suppose $g(y')\downarrow$ for all $y' < y$. Then from the sequence $(g(y'))_{y' < y}$ one can effectively decide whether or not $y < \Delta\mathcal{O}(g,x)$: Feed the sequence $(g(y'))_{y' < y}$ into $\mathcal{D}_{\mathcal{O}}$ and wait until either $\mathcal{D}_{\mathcal{O}}$ produces $\mathcal{O}(g,x)$ or $\mathcal{D}_{\mathcal{O}}$ requests $g(y)$ before producing $\mathcal{O}(g,x)$. One or the other must happen since $\mathcal{O}: R \to R$.[1] In the first case $y \ge \Delta\mathcal{O}(g,x)$

[1]Without loss of generality we assume that if $\mathcal{D}_{\mathcal{O}}$ make a request then it does nothing until it gets a new input.

while in the second, $y < \Delta\mathcal{O}(g,x)$.

The following lemma is needed in a patching argument. In a first pass through this proof of 1, we suggest the reader bypass this lemma and try to understand the construction of M with $\mathcal{O}(\phi_j, \max\{i,j,x\})$ in place of $\mathcal{O}(\phi_{\sigma(i,j,x)}, \sigma(i,j,x))$. The only error then occurs in the next to last line of the proof of 1, viz. "M[f] will converge at least by stage i."

Lemma: Let $\mathcal{O}:R \to R$ be a recursive operator. Then there is a recursive function σ, uniform in $\mathcal{O}$, such that for all i, j, x, $\sigma(i,j,x) \geq \max\{i,j,x\}$, and for all i, j, x and all $f \in R$, if $\phi_j(y) = f(y)$ for all $y < \Delta\mathcal{O}(\phi_{\sigma(i,j,x)}, \sigma(i,j,x))$

then (1) $\mathcal{O}(\phi_{\sigma(i,j,x)}, \sigma(i,j,x)) \downarrow$

and (2) $\phi_i = f \Rightarrow \phi_{\sigma(i,j,x)} = f$.

Proof of lemma: $\sigma(i,j,x)$ is defined by implicit use of the recursion theorem and padding (to ensure $\sigma(i,j,x) \geq \max\{i,j,x\}$) to be an index of the following algorithm:

$\phi_{\sigma(i,j,x)}(y)$ = "Suppose $\phi_{\sigma(i,j,x)}(y')$ has been defined for all $y' < y$. We now define $\phi_{\sigma(i,j,x)}(y)$: By Remark (iii) above, we can decide whether or not $y < \Delta\mathcal{O}(\phi_{\sigma(i,j,x)}, \sigma(i,j,x))$.

case (a) Suppose $y < \Delta\mathcal{O}(\phi_{\sigma(i,j,x)}, \sigma(i,j,x))$. Then set $\phi_{\sigma(i,j,x)}(y) = \phi_j(y)$.

case (b) Suppose $y \geq \Delta\mathcal{O}(\phi_{\sigma(i,j,x)}, \sigma(i,j,x))$. Then set $\phi_{\sigma(i,j,x)}(y) = \phi_i(y)$."

By the effectiveness of the recursion theorem, $\underline{\sigma \text{ is recursive}}$ and $\underline{\sigma \text{ is uniform in } \mathcal{O}}$. By construction $\underline{\sigma(i,j,x) \geq \max\{i,j,x\}}$.
By the construction of σ, it follows that if $\phi_j(y) = f(y)$ for all $y < \Delta\mathcal{O}(\Phi_{\sigma(i,j,x)}, \sigma(i,j,x))$

then (1) $\phi_{\sigma(i,j,x)}(y) = \phi_j(y)$ for all $y < \Delta\mathcal{O}(\Phi_{\sigma(i,j,x)}, \sigma(i,j,x))$.

Hence, $\Phi_{\sigma(i,j,x)}(y) \downarrow$ for all $y < \Delta\mathcal{O}(\Phi_{\sigma(i,j,x)}, \sigma(i,j,x))$.

Hence $\mathcal{O}(\Phi_{\sigma(i,j,x)}, \sigma(i,j,x)) \downarrow$.

and (2) If $\phi_i = f$ then $\phi_{\sigma(i,j,x)}(y) = \phi_i(y)$ for all $y \geq \Delta\mathcal{O}(\Phi_{\sigma(i,j,x)}, \sigma(i,j,x))$. So $\phi_{\sigma(i,j,x)} = f$.

¤

Construction of M:

M[f] = "Go to stage 0.

<u>Stage i</u>: Output i. Spend half your time doing A and half doing B:

A. Look for an x such that $\phi_i(x) \downarrow$ and $\phi_i(x) \neq f(x)$. If and when such an x is found, go to stage i+1.

B. For every pair (j,x) such that $\mathcal{O}(\Phi_{\sigma(i,j,x)}, \sigma(i,j,x))\downarrow$ and such that $\phi_j(y) = f(y)$ for all $y < \Delta\mathcal{O}(\Phi_{\sigma(i,j,x)}, \sigma(i,j,x))$ check that $\Phi_i(x) \leq \mathcal{O}(\Phi_{\sigma(i,j,x)}, \sigma(i,j,x))$. If this check fails for some (j,x), go to stage i+1."

Clearly, $\underline{\text{M is uniform in } \mathcal{O}}$.

Let $f \in R$ have the property that M[f] converges, say to i. Thus M[f] never gets past stage i. Let $\phi_j = f$. Then (cf. part (1) of lemma) $\mathcal{O}(\Phi_{\sigma(i,j,x)}, \sigma(i,j,x)) \downarrow$ for all x. Hence, $\Delta\mathcal{O}(\Phi_{\sigma(i,j,x)}, \sigma(i,j,x)) < \infty$ and so M in stage i(B) will verify $\phi_j(y) = f(y)$ for all $y < \Delta\mathcal{O}(\Phi_{\sigma(i,j,x)}, \sigma(i,j,x))$. It follows that M in stage i(B) will verify that $\Phi_i(x) \downarrow$

for all x. As a consequence, M in stage i(A) will verify that $\phi_i(x) = f(x)$ for all x. Hence M[f] converges to an index for f. It follows that <u>M is strong on R</u>.

Finally, let $f \in R$ and suppose that f is everywhere $\mathcal{O}$-compressed with compression index i. Then M[f] will converge at least by stage i.[1] Since M is strong on R, it follows that <u>M can identify f</u>.

<u>2.</u> We are given a machine M that is strong on R. From M it is easy to construct a machine M' that is strong on R, that is as powerful as M on R, that initially outputs index 0, and that has the property that after n steps of computation, if M' outputs an index i_n, then $i_n \geq n$. (Thus M' can diverge only by changing its mind infinitely often and can converge only by outputing a finite number of integers.) From now on we shall write M in place of M'. Notice that M is not order-independent.

Define $\mathcal{O}$ as follows:

$\mathcal{O}(g,m)$ = "By dovetailing, search for all integers $j \leq m$ such that $g(y) \neq \Phi_j(y)$ for some $y \in N$. For each such j that is found, cancel j and set $\hat{j} = 0$.

While the above procedure is going on, do the following for each uncancelled j, $j \leq m$, for as long as that j remains uncancelled: Compute exactly m steps of $M[\vec{\phi}_j]$ for the increasing enumeration $\vec{\phi}_j$ of ϕ_j. Let j_m denote the last integer that $M[\vec{\phi}_j]$ produced during this m-step computation. Next,

---

[1]This is not true if $\mathcal{O}(\Phi_j, \max\{i,j,x\})$ is substituted for $\mathcal{O}(\Phi_{\sigma(i,j,x)}, \sigma(i,j,x))$ in the construction of M.

The reason M[f] <u>will</u> converge by stage i is this: Suppose $\overline{\mathcal{O}(\Phi_{\sigma(i,j,x)}, \sigma(i,j,x))} \downarrow$ and $\phi_j(y) = f(y)$ for all $y < \Delta\mathcal{O}(\Phi_{\sigma(i,j,x)}, \sigma(i,j,x))$. Then (cf. part (2) of lemma) $\phi_{\sigma(i,j,x)} = f$ and so $\Phi_i(x) \leq \mathcal{O}(\Phi_{\sigma(i,j,x)}, \max\{i, \sigma(i,j,x), x\})$. But $\sigma(i,j,x) \geq \max\{i,j,x\}$. So $\Phi_i(x) \leq \mathcal{O}(\Phi_{\sigma(i,j,x)}, \sigma(i,j,x))$.

It follows from the proof of 2 that the set of recursive functions that are everywhere $\mathcal{O}$-compressed for some recursive operator $\mathcal{O}: R \to R$ are actually everywhere $\mathcal{O}'$-compressed for some general recursive operator $\mathcal{O}'$. This suggests that relatively "simple" operators are all that is needed to construct inductive inference machines of great power.

As we shall see from applications 1 and 2, the particular operator $\mathcal{O}(f) = \lambda x[\max\{f(0), \ldots, f(2x)\}]$ yields an exceptionally powerful machine. Paul Young has pointed out to us that the machine $M_{\mathcal{O}}$ defined by this operator converges when fed a Martin-Löf [ ] random total function. To see this, use the fact that if ϕ_j is consistent with a Martin-Löf random function f (i.e., $\phi_j(x) \downarrow$ and $f(x) \downarrow \Rightarrow \phi_j(x) = f(x)$), then $\phi_j(x)$ may not converge on much more than $\log_2 j$ integers x. Now, when $M_{\mathcal{O}}$ is fed a Martin-Löf random f, it conjectures and eliminates inconsistent indices until it finally conjectures an index i such that ϕ_i is consistent with f. Furthermore, once $M_{\mathcal{O}}$ conjectures this i, it never finds a reasonable alternate hypothesis j to replace it. This is because no algorithm j can correctly compute f(0), ..., f(2j) for a Martin-Löf random f. Hence $M_{\mathcal{O}}$ converges to i. Here then is a curious distinction between the machines of Theorem 3 and those of Theorem 4: The machines of Theorem 3 necessarily diverge on Martin-Löf random functions while those of Theorem 4 do not.

6. Applications

1. Let σ,h be two recursive functions. An inductive inference machine strong on R can be designed to identify every 0-1 valued recursive function f having a program i such that

(a) Φ_i is monotonically increasing, and continue the computation of $M[\vec{\phi}_j]$ and simultaneously start computing $\phi_{j_m}(0), \ldots, \phi_{j_m}(m)$. If $M[\vec{\phi}_j]$ produces a new conjecture (<u>after</u> m steps), then cancel j and set $\hat{j} = 0$. On the other hand, if $\phi_{j_m}(0), \ldots, \phi_{j_m}(m)$ all converge, set $\hat{j} = \max\{\Phi_{j_m}(0), \ldots, \Phi_{j_m}(m)\}$.

All of the above procedures are stopped once $\hat{j}$ is defined for every $j \leq m$. At that time, set $\mathcal{O}(g,m) = \max\{\hat{j} \mid j \leq m\}$."

Clearly, <u>$\mathcal{O}$ is uniform in M</u>.

<u>$\mathcal{O}$ is a general recursive operator</u>[1]: Let g be a total function and let $m \in N$. We want to show that $\mathcal{O}(g,m) \downarrow$. Assume to the contrary that the algorithm for $\mathcal{O}(g,m)$ never defines $\hat{j}$ for some $j \leq m$. Then $g = \Phi_j$ and $M[\vec{\phi}_j] \downarrow j_m$. In this case, M strong on $R \Rightarrow (\phi_{j_m} = \phi_j) \Rightarrow \phi_{j_m}$ total $\Rightarrow \max\{\Phi_{j_m}(0), \ldots, \Phi_{j_m}(m)\} \downarrow$. This means that $\hat{j}$ is defined, which is a contradiction.

<u>$\mathcal{O}: R \to R$</u> because $\mathcal{O}$ is general recursive.

Let $f \in R$. <u>M can identify $f \Rightarrow f$ is everywhere $\mathcal{O}$-compressed</u>: Suppose $M[\vec{f}] \downarrow i$($\phi_i = f$ since M is strong on R). Suppose $\phi_j = f$ (thus $M[\vec{\phi}_j] \downarrow i$), $x \in N$, and $m = \max\{i,j,x\}$. We are required to prove that $\mathcal{O}(\Phi_j,m) \geq \Phi_i(x)$: Notice that the device for computing $\mathcal{O}(\Phi_j,m)$ will attempt to cancel j on the grounds that $\Phi_j \neq \Phi_j$, but naturally this will fail. Hence it will compute $M[\vec{\phi}_j]$ for m steps, thereby producing j_m. $M[\vec{\phi}_j]$ will never change its mind after producing j_m (for if it did then i would be greater than m, according to our construction of M', and this would contradict the definition of m). Hence $j_m = i$ and $\hat{j} = \max\{\Phi_{j_m}(0), \ldots, \Phi_{j_m}(m)\} = \max\{\Phi_i(0), \ldots, \Phi_i(m)\}$. Therefore, $\mathcal{O}(\Phi_j,m) \geq \hat{j} \geq \Phi_i(x)$, as was to be proved. ■

[1]A general recursive operator $\mathcal{O}$ is a recursive operator that maps all total functions into total functions.

(b) for every program j for f, $\Phi_i(x) \leq h(x,\Phi_j(x))$ for all but (at most) $\sigma(j)$ integers x.

The compression theorem [ ] supplies arbitrarily difficult 0-1 valued recursive functions f satisfying the above requirements[1] (with $\sigma = \lambda x[x]$ and, in the case of Turing machines, $h(x,y) = (x+y)^6$.

An operator for which these functions are everywhere $\mathcal{O}$-compressed is given by $\mathcal{O}(g,m) = \max\{h(m+n,g(m+n)) \mid n \in [0, \max_{k<m} \sigma(k)]\}$. To see this note that if f and its program i satisfy (b) and if j is any program for f, then for each m there is an $n \in [0,\sigma(j)]$ such that $\Phi_i(m+n) \leq h(m+n, \Phi_j(m+n))$. Hence for all x and for $m = \max\{i,j,x\}$,

$$\begin{aligned}\mathcal{O}(\Phi_j,m) &= \max\{h(m+n, \Phi_j(m+n)) \mid n \in [0, \max_{k\leq m} \sigma(k)]\} \\ &\geq \Phi_i(m+n) \text{ for some } n \in [0,\sigma(j)] \\ &\geq \Phi_i(m) \geq \Phi_i(x) \quad \text{if } i \text{ satisfies (a).}\end{aligned}$$

Hence i is a compression index for f with respect to $\mathcal{O}$. A refinement of this argument shows that there exist arbitrarily difficult 0-1 valued recursive functions f that are everywhere $\mathcal{O}$-compressed for such simple $\mathcal{O}$ as $\mathcal{O}(g) = \lambda x[\max\{g(0), \ldots, g(2x)\}]$.

2. Define $\tau \in R$ so that

$$\phi_{\tau(i)}(x) = \begin{cases} 1 & \text{if } \Phi_i(x)\downarrow \text{ and } \Phi_x(x) \leq \Phi_i(x) \\ 0 & \text{if } \Phi_i(x)\downarrow \text{ and } \Phi_x(x) > \Phi_i(x) \\ \uparrow & \text{if } \Phi_i(x)\uparrow \end{cases}$$

For $\Phi_i \in R$, $\phi_{\tau(i)}$ may be viewed as providing an approximation to the halting problem. The bigger Φ_i is, the better is the approximation provided by $\phi_{\tau(i)}$. As we shall now show, the set $\{\phi_{\tau(i)} \mid \Phi_i$ is a monotonically increasing total function$\}$ can be identified:

[1] This shows that machines strong on R are strictly more powerful than machines strong on the set of all total functions, F. Why? Because for every machine M that is strong on F there is a recursive function h such that the 0-1 valued recursive functions that M can identify are all h-honest. This follows from a binary-tree argument (König's lemma).

Proposition There exists a general recursive operator $\mathcal{O}$ such that for every monotonically increasing $\Phi_i \in R$, $\phi_{\tau(i)}$ is everywhere $\mathcal{O}$-compressed.

Lemma 1 There exists a $q \in R$ monotonically increasing in all three variables with the property that for every i and every x in the domain of ϕ_i,

$$\Phi_{\tau(i)}(x) \leq q(i,x,\Phi_i(x)).$$

□

Lemma 2 There exists $p, \sigma \in R$ such that for every j and m, $\sigma(j,m) \geq \max\{j,m\}$ and

(i) $\phi_{\sigma(j,m)}(\sigma(j,m)) \downarrow \iff \phi_j(\sigma(j,m)) = 0$

(ii) $\Phi_{\sigma(j,m)}(\sigma(j,m)) \leq p(j,m,\Phi_j(\sigma(j,m)))$ if $\phi_j(\sigma(j,m)) = 0$.

Proof of lemma Define $\sigma \in R$ so that $\sigma(j,m) \geq \max\{j,m\}$ and

$$\phi_{\sigma(j,m)}(x) = \begin{cases} 0 \text{ if } \phi_j(\sigma(j,m)) \downarrow 0 \text{ (implicit recursion Theorem)} \\ \uparrow \text{ otherwise} \end{cases}$$

Set $p(j,m,z)$ = "If $\Phi_j(\sigma(j,m)) = z$ and $\phi_j(\sigma(j,m)) = 0$, output $\Phi_{\sigma(j,m)}(\sigma(j,m))$. Otherwise, output 0." □

Proof of proposition Set $\mathcal{O}(g,m) = \max\{q(i,m,p(j,m,g(\sigma(j,m)))) \mid i,j \leq m\}$ Since $q,p,\sigma \in R$, $\mathcal{O}$ is a general recursive operator.

Now suppose $\Phi_i \in R$ is monotonically increasing. Let $\phi_j = \phi_{\tau(i)}$, let $x \in N$, and let $m = \max\{\tau(i),i,j,x\}$. We show that $\Phi_{\tau(i)}(x) \leq \mathcal{O}(\Phi_j,m)$: First note that

$$\phi_{\sigma(j,m)}(\sigma(j,m)) \downarrow \Rightarrow \phi_j(\sigma(j,m)) = 0 \text{ by lemma 2(i), and}$$

$$\phi_{\sigma(j,m)}(\sigma(j,m)) \uparrow \Rightarrow \phi_{\tau(i)}(\sigma(j,m)) = 0 \text{ by definition of } \phi_{\tau(i)}.$$

Since $\phi_j = \phi_{\tau(i)}$, the above two equations imply that

$$\phi_{\tau(i)}(\sigma(j,m)) = \phi_j(\sigma(j,m)) = 0.$$

Hence $\mathcal{O}(\Phi_j,m) \geq q(i,m,p(j,m,\Phi_j(\sigma(j,m))))$ by definition of $\mathcal{O}$

$\geq q(i,x,\Phi_i(x))$ by the monotonicity of q and the fact

that $p(j,m,\Phi_j(\sigma(j,m))) \geq \Phi_{\sigma(j,m)}(\sigma(j,m))$ by lemma 2(ii)

and the fact that $\phi_j(\sigma(j,m)) = 0$

$\geq \Phi_i(\sigma(j,m))$ by definition of $\phi_{\tau(i)}$

and the fact that

$\phi_{\tau(i)}(\sigma(j,m)) = 0$

$\geq \Phi_i(x)$ since $\sigma(j,m) \geq m \geq x$

and Φ_i is monotonically

increasing.

$\geq \Phi_{\tau(i)}(x)$ by lemma 1. ∎

Suppose that $(\phi_i)_{i \in N}$ is a standard acceptable Gödel numbering defined in terms of Turing machines with binary input-output code. Let $\mathcal{O}$ be the recursive operator defined by $\mathcal{O}(g) = \lambda x[\max\{g(0), \ldots, g(2x)\}]$. In this special case, then, $\phi_{\tau(i)}$ is everywhere $\mathcal{O}$-compressed for every $\Phi_i \in R$ that is monotonically increasing <u>and</u> rapidly growing. We omit details.

3. As Albert Meyer has pointed out to us, there exists a recursive operator $\mathcal{O}$: $R \to R$ with the property that for every $h \in R_2$, a 0-1 valued recursive function f can be constructed that is everywhere $\mathcal{O}$-compressed and has h-speedup almost everywhere. Thus some arbitrarily speedable functions can be identified by machine.

4. Let $h,\sigma \in R$. There exists a machine M strong on R that can identify all recursive functions f having the property that

$$\exists i[\phi_i = f \text{ and } \forall_j > i\ [\phi_j = f \Rightarrow \forall x > i\ [\Phi_i(x) < \sum_{z=0}^{\sigma(j,x)} h(z,\Phi_j(z))]]]$$

This operator-free formula conveys much but not all of the power available to machines that are strong on R.

We now give another quite different characterization of the recursive functions that can be identified by a machine strong on R. This theorem and its proof are very close to Theorem 3, which characterizes the machines strong on P.

Definition Let $\mathcal{O}: R \to R$ be a recursive operator. We say that $f \in R$ is $\mathcal{O}$-honest iff $\exists i[\phi_i = f$ and $\Phi_i(x) \leq \mathcal{O}(f,x)$ almost everywhere].

Theorem 5 Let M denote an inductive inference machine that is strong on R. Let $\mathcal{O}$ denote a recursive operator that maps R into R.

1. For every $\mathcal{O}$ there is an M uniform in $\mathcal{O}$ such that for all $f \in R$, f is $\mathcal{O}$-honest $\Rightarrow$ M can identify f.

2. For every M there is an $\mathcal{O}$ uniform in M such that for all $f \in R$, M can identify f $\Rightarrow$ f is $\mathcal{O}$-honest.

Proof 1. First note that if $f \in R$ is $\mathcal{O}$-honest, then there exist integers i,k such that $\phi_i = f$ and $\Phi_i(x) \leq \max\{k, \mathcal{O}(f,x)\}$ for all x. This suggests the following definition of M:

M[f] = "Go to stage 1.

Stage $n = \langle i,k \rangle$: Output i. Check that $\Phi_i(x) \leq \max\{k,\mathcal{O}(f,x)\}$ and that $\phi_i(x) = f(x)$ for x = 0, 1, 2, ... As soon as this check fails for some x, go to stage n + 1."

Clearly, M is uniform in $\mathcal{O}$.

M is strong on R: Suppose $f \in R$ and that $M[f] \downarrow i$. Since M[f] changes its mind (i.e., outputs a different integer) each time it changes stage, it follows that M[f] must eventually enter and never leave some stage $n = \langle i,k \rangle$. In this stage M verifies that $\phi_i = f$.

<u>f is $\mathcal{O}$-honest $\Rightarrow$ M can identify f</u>: Suppose $f \in R$ is $\mathcal{O}$-honest. Choose i,k so that $\phi_i = f$ and $\Phi_i(x) \leq \max\{k, \mathcal{O}(f,x)\}$ for all x. Then M[f] can never go past stage $\langle i,k \rangle$. It follows that M[f] will converge, and by the strongness of M, will converge correctly.

<u>2</u>. $\mathcal{O}(f,x)$ = "Feed f in increasing order $(0,f(0))$, $(1,f(1))$, ... to M until M requests its x + 1st input. Let i_x denote the last output produced by M at that time. Now continue the computation of M[f] and in addition start computing $\phi_{i_x}(x)$. Do both until either

(a) M changes its mind about i_x, in which case output 0, or

(b) $\phi_{i_x}(x) \downarrow$, in which case output $\Phi_{i_x}(x)$."

Clearly, <u>$\mathcal{O}$ is uniform in M</u> and <u>$\mathcal{O}$ is a recursive operator</u>.

Now suppose $f \in R$. We want to show that $\mathcal{O}(f,x) \downarrow$: Since M is strong on R, M[f] will either change its mind infinitely often or else will converge on an index for f. If M converges by the time it requests the x + 1st input, then i_x is an index for f, whence $\phi_{i_x}(x) \downarrow$, whence $\mathcal{O}(f,x)$ is defined in (b). If, on the other hand, i_x is not the last output produced by M[f], then $\mathcal{O}(f,x)$ will eventually halt, in (a) if not (b). Hence <u>$f \in R \Rightarrow \mathcal{O}(f) \in R$</u>.

Let $f \in R$. We now show that <u>M can identify f $\Rightarrow$ f is $\mathcal{O}$-honest</u>: Choose n such that M[f] converges by the time it requests the n + 1st input. Then $\phi_{i_n} = f$ and $\mathcal{O}(f,x) = \Phi_{i_n}(x)$ for all $x \geq n$. ■

Theorems 4 and 5 are very different: For example, the operator of Theorem 4 can always be chosen to be general recursive, but not so the operator of Theorem 5.

Together, Theorems 4 and 5 clarify an open question in abstract complexity theory about what are the operator honest recursive functions:

Corollary The operator honest recursive functions are precisely the everywhere operator compressed recursive functions (i.e., for every operator $\mathcal{O}:R \rightarrow R$ there is an operator $\mathcal{O}':R \rightarrow R$ such that (1) all $\mathcal{O}$-honest recursive functions are everywhere $\mathcal{O}'$-compressed, and (2) all everywhere $\mathcal{O}$-compressed recursive functions are $\mathcal{O}'$-honest).

This characterization really does give insight about what it means for a function to be $\mathcal{O}$-honest. For example, see the applications at the beginning of this section.

7. Strong Identification on P_∞

Let P_∞ denote the set of all partial recursive functions with infinite domain. Machines strong on P are weak only because they must infer all finite functions. Machines strong on P_∞, on the other hand, can infer many though not all of the recursive functions that can be infered by machines strong on R. For example, the set of recursive functions mentioned in application 1 of section 6 can be inferred by a machine strong on P_∞, but not so the set mentioned in application 2.

8. Open Problems

① Say that M can identify f almost everywhere iff whenever M is fed f in any order whatsoever, then it eventually converges on an index for a partial recursive function ϕ_i that extends f almost everywhere, i.e., $\phi_i(x) = f(x)$ for all but a finite number of integers $x \in$ domain (f). Let S be a set of partial recursive functions. Say that M is strong on S with respect to almost everywhere identification iff whenever M[f] converges, say to i, for some $f \in S$, then ϕ_i is almost everywhere an extension of f. An open problem is to characterize the set of partial recursive functions that can be identified by a machine that is strong on P(or R) with respect to almost everywhere identification.

② Characterize the sets of partial or total recursive functions that can be identified by arbitrary (not necessarily strong) inductive inference machines.

9. Conclusion

Our main result, theorem 4 part 1, is an algorithm schema for designing powerful inductive inference machines. In the usual recursion-theoretic manner, our machines are designed to enumerate all hypotheses, good and bad, in order to select the most likely hypothesis for a given input sequence.

A weakness of our machines lies with this enumeration process: In practice, inductive inference machines should be designed to generate reasonably good hypotheses right from the start, and then to select the best hypothesis from among them. Our machines actually generate all hypotheses, not just the reasonably good ones, and so they are absurdly inefficient.

A strength of our machines, on the other hand, lies in the particular and unusual criteria they employ to determine which of several hypotheses is best. By these criteria, a machine does not reject a hypothesis on the naive grounds that it is computationally difficult. Instead the criteria actually defend a machine's latest hypothesis, as long as that hypothesis explains approximately as much data in a given amount of time as any competing hypothesis.

Theorem 4 part 2 asserts that the above-mentioned criteria are in a sense the best possible. Because of this, we expect that they will eventually be employed in the design of practical inductive inference machines.

Acknowledgements

Our early study of inductive inference took place during the fall of '71 at the Artificial Intelligence Group, MIT, under the sponsorship of Marvin Minsky and Seymour Papert. We thank them and Albert Meyer for their welcome encouragement.

The main impetus to this work was the discovery and proof of what is now application 1 to Theorem 4 during the summer of '72 at Berkeley. Theorem 2 and a number of other results not mentioned here were also obtained at this time.

Further discussion of the formal problem took place during the fall of '72 at the Cybernetics group of Guiseppe Trautteur, The University of Naples, Italy. Special credit is due to Eliana Minicozzi of that group for her study of strong machines and her discovery and proof of the Union Theorem. More of her results appear in [ ].

The discovery and proof of Theorem 4 is work that the authors did in the winter of '72 in Caracas. The proof of Theorem 5 came shortly later in Berkeley.

References

1. Blum, M., A Machine-Independent Theory of the Complexity of Recursive Functions. J. ACM 14, 2 (April 1967), 322-336.

2. Feldman, J., Some Decidability Results on Grammatical Inference and Complexity, Information and Control 20, No. 3, (April 1972), 244-262.

3. Gold, E. M., Language Identification in the Limit. Information and Control 10, (1967), 447-474.

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

5. Meyer, A. R. and Fischer, P. C., Computational Speed-up by Effective Operators. J. Symbolic Logic 37, 1 (March 1972), 55-68.

6. Minicozzi, E., Some Natural Properties of Inductive Inference. Istituto di Fisica Teorica, Naples, Italy; to be published.

7. Rogers, H., Jr., Recursive Functions and Effective Computability. McGraw-Hill, New York, 1967.

8. Solomonoff, R., A Formal Theory of Inductive Inference. Information and Control 7, (1964), 1-22, 224-254.