Costs of General Purpose Learning

John Case [a] Keh-Jiann Chen [b] Sanjay Jain [c]

[a] *Department of CIS*
University of Delaware
Newark, DE 19716
USA
Email: case@cis.udel.edu

[b] *Institute for Information Sciences*
Academica Sinica
Taipei, 15, Taiwan
Republic of China

[c] *School of Computing*
National University of Singapore
Singapore 119260
Email: sanjay@comp.nus.edu.sg

Abstract

Leo Harrington surprisingly constructed a machine which can learn *any* computable function f according to the following criterion (called $\mathbf{Bc}^*$-*identification*). His machine, on the successive graph points of f, outputs a corresponding infinite sequence of programs $p_0, p_1, p_2, \ldots$, and, for some i, the programs $p_i, p_{i+1}, p_{i+2}, \ldots$ each compute a variant of f which differs from f at only finitely many argument places. A machine with this property is called *general purpose*. The sequence $p_i, p_{i+1}, p_{i+2}, \ldots$ is called a *final sequence*.

For Harrington's general purpose machine, for distinct m and n, the finitely many argument places where p_{i+m} fails to compute f can be very different from the finitely many argument places where p_{i+n} fails to compute f. One would hope though, that if Harrington's machine, or an improvement thereof, inferred the program p_{i+m} based on the data points $f(0), f(1), \ldots, f(k)$, then p_{i+m} would make very few mistakes computing f at the "near future" arguments $k+1, k+2, \ldots, k+\ell$, where ℓ is reasonably large. Ideally, p_{i+m}'s finitely many mistakes or anomalies would (mostly) occur at arguments $x \gg k$, i.e., ideally, its anomalies would be well placed beyond near future arguments. In the present paper, for general purpose learning machines, it is analyzed just how well or badly placed these anomalies may be with respect to near future arguments and what are the various tradeoffs.

In particular, there is good news and bad. Bad news is that, for any learning machine $\mathbf{M}$ (including general purpose $\mathbf{M}$), for all m, there exist infinitely many computable functions f such that, infinitely often $\mathbf{M}$ incorrectly predicts f's next m

Preprint submitted to Elsevier Preprint *2 October 1999*

near future values. Good news is that, for a suitably clever general purpose learning machine $\mathbf{M}$, for each computable f, for $\mathbf{M}$ on f, the *density* of any such associated bad prediction intervals of size m is vanishingly small.

Considered too is the possibility of providing a general purpose learner which *additionally* learns some interesting classes with respect to much stricter criteria than $\mathbf{Bc}^*$-identification. Again there is good news and bad. The criterion of *finite identification* requires for success that a learner $\mathbf{M}$ on a function f output exactly one program which correctly computes f. $\mathbf{Bc}^n$*-identification* is just like $\mathbf{Bc}^*$-identification above except that the number of anomalies in each program of a final sequence is $\leq n$. Bad news is that there is a finitely identifiable class of computable functions $\mathcal{C}$ such that for *no* general purpose learner $\mathbf{M}$ and for no n, does $\mathbf{M}$ *additionally* $\mathbf{Bc}^n$-identify $\mathcal{C}$. $\mathbf{Ex}$*-identification* by $\mathbf{M}$ on f requires that $\mathbf{M}$ on f converges, after a few output programs, to a *single* final program which computes f. A *reliable* learner (by definition) never deceives by false convergence; more precisely: whenever it converges to a final program on a function f, it must $\mathbf{Ex}$-identify f. Good news is that, for any class $\mathcal{C}$ that can be *reliably* $\mathbf{Ex}$-identified, there is a general purpose machine which *additionally* $\mathbf{Ex}$-identifies $\mathcal{C}$!

1 Introduction

The learning situation often studied in inductive inference [JORS99] may be described as follows. A learner receives as input, one at a time, the successive graph points of a function f. As the learner is receiving its input, it conjectures a sequence of programs as hypotheses. To be able to learn the function f, the sequence of programs conjectured by the learner must have some desirable relation to the input function f. By appropriately choosing this desirable relation one gets different criteria of successful learning. One of the first such criteria studied is called $\mathbf{Ex}$*-identification* ([Gol67,BB75,CS83]). The learner is said to $\mathbf{Ex}$*-identify a function* f iff the sequence of programs output by it on f, after a few output programs, converges to a single final program which computes f.[1] A learner is said to $\mathbf{Ex}$*-identify a class* iff it $\mathbf{Ex}$-identifies each function in the class. A *class of functions is* $\mathbf{Ex}$*-identifiable* iff some machine $\mathbf{Ex}$-identifies the class.

Even though one cannot $\mathbf{Ex}$-identify the class of all the computable functions [Gol67], there are large and useful classes of functions which can be $\mathbf{Ex}$-identified. For example, any recursively enumerable class of computable functions such as the class of polynomials or the class of primitive recursive functions [Rog67] is $\mathbf{Ex}$-identifiable.

[Bār74,CS83] considered a generalization of $\mathbf{Ex}$-identification called $\mathbf{Bc}$-

[1] In general more formal definitions are in Section 2 below.

identification. In **Bc**-identification of a function f by a machine $\mathbf{M}$ one requires that the sequence of programs output by $\mathbf{M}$ on f either converges to a program for f, or the sequence of programs is infinite, with all but finitely many of them being (possibly different) programs for f. [CS83] also considered the variants of the **Ex** and **Bc**-identification criteria in which the final programs need not be perfect, but are allowed to have some anomalies or mistakes in their predictions of I/O behavior. For n a natural number, if the final programs are allowed to make at most n errors, then the criteria of inference are called $\mathbf{Ex}^n$ and $\mathbf{Bc}^n$ respectively. If the final programs are allowed to make at most finitely many errors, then the criteria of inference are called $\mathbf{Ex}^*$ and $\mathbf{Bc}^*$ respectively.

Harrington [CS83] constructed a machine which $\mathbf{Bc}^*$-identifies each computable function! In the present paper, we call machines which do this *general purpose*. However, on infinitely many computable functions, the final programs output by Harrington's machine become more and more degenerate, i.e., the finite sets of anomalies in successive final output programs, in general, grow in size without bound. We note that this is a property of any general purpose learner, and, in fact, the number of anomalies grows faster than any computable bound (Theorem 4 in Section 3 below).

Since the programs output by any general purpose learning machine make large numbers of mistakes (on infinitely many computable functions), it would be interesting to study how these errors are distributed. For example, in real life one probably cares more about "near future errors" than "distant future errors". Based on this motivation in Section 4 below we define new criteria of inference called $\mathbf{Bc}^n_m$. Informally, for a machine to $\mathbf{Bc}^n_m$-*identify* a function f, for its final programs, their predictions on the next m inputs should have at most n errors. In Section 4 we completely resolve the relationship between different $\mathbf{Bc}^n_m$ criteria of inference (Corollary 23 in Section 4). In particular, we show that for any learning machine $\mathbf{M}$, (including general purpose $\mathbf{M}$), for all m, there exist infinitely many computable functions f such that, infinitely often $\mathbf{M}$ incorrectly predicts f's next m near future values (Corollary 24)! Thus there is an ostensibly unpleasant cost to general purpose learning. As we will see, though, this can, be assuaged at least in some interesting respects described below.

In contrast to the result mentioned above that any general purpose learning machine $\mathbf{M}$ predicts next m values wrongly infinitely often, we show that the *density* of such bad prediction intervals can be made very small (Theorem 28 in Section 5 below).

A *reliable* learner (by definition) never deceives by false convergence; more precisely: whenever it converges to a final program on a function f, it must **Ex**-identify f [Min76,BB75,CJNM94]. For example, r.e. classes of computable functions (such as the class of polynomial functions and the class of primitive recursive functions [Rog67]) as well as the class of total run time functions can be reliably **Ex**-identified [BB75,CS83]. On a further positive note, we show that for every reliably **Ex**-identifiable class of computable functions $\mathcal{S}$, there is a general purpose learning machine which **Ex**-identifies $\mathcal{S}$ (Theorem 30 in Section 5 below)!

The criterion of *finite identification* requires for success that a learner $\mathbf{M}$ on a function f output exactly one program which correctly computes f. Learning by finite identification can be thought of as *one-shot learning.* We show, by contrast to the result in the immediately above paragraph (Theorem 30), that there is a class $\mathcal{S}$ which *is* finitely identifiable, yet for all n, *no* general purpose learner can additionally $\mathbf{Bc}^n$-identify $\mathcal{S}$ (Corollary 36 in Section 5 below).

Freivalds and Wiehagen [FW79] showed that there exists a machine which can identify all the recursive functions if, in addition to the graph of the input function, it is given an arbitrary upper bound on the size of the minimal program computing the input function as additional information. Freivalds, Botuscharov and Wiehagen [FBW98] further showed that in some (but not all) acceptable programming systems, the above machine can produce a final program of size within the upper bound given as additional information. Machines, as in above, exhibit a different kind of general purpose behaviour. We will not deal with above type of general purpose learners in this paper.

We now proceed formally.

2 Notation and Preliminaries

Recursion-theoretic concepts not explained below are treated in [Rog67]. N denotes the set of natural numbers. $*$ denotes a non-member of N and is assumed to satisfy $(\forall n \in N)[n < * < \infty]$. Let $\in, \subseteq, \subset, \supseteq, \supset$, respectively denote membership, subset, proper subset, superset and proper superset relations for sets. Emptyset is denoted by $\emptyset$. Cardinality of a set S is denoted by $\mathrm{card}(S)$. So "$\mathrm{card}(S) \leq *$" means that $\mathrm{card}(S)$ is finite. We let $\min(S)$ and $\max(S)$, respectively, denote the minimum and maximum element in S. We take $\min(\emptyset)$ to be ∞ and $\max(\emptyset)$ to be 0.

$\langle \cdot, \cdot \rangle$ denotes a 1-1 computable mapping from pairs of natural numbers onto natural numbers. π_1, π_2 are the corresponding projection functions. $\langle \cdot, \cdot \rangle$ is extended to n-tuples in a natural way.

Λ denotes the empty function. η, with or without decorations (decorations are subscripts, superscripts, primes and such), ranges over partial functions.

$\eta(x)\downarrow$ denotes that $\eta(x)$ is defined. $\eta(x)\uparrow$ denotes that $\eta(x)$ is not defined. For $a \in N \cup \{*\}$, $\eta_1 =^a \eta_2$ means that $\mathrm{card}(\{x \mid \eta_1(x) \neq \eta_2(x)\}) \leq a$. $\eta_1 \neq^a \eta_2$ means that $\neg[\eta_1 =^a \eta_2]$. (If η_1 and η_2 are both undefined on input x, then, as is standard, we take $\eta_1(x) = \eta_2(x)$.) If $\eta =^a f$, then we often call a program for η as an a-error program for f. We let domain(η) and range(η) respectively denote the domain and range of the partial function η.

We let f, g and h, with or without decorations, range over total functions. $\mathcal{R}$ denotes the class of all *computable* functions, i.e., total computable functions with arguments and values from N. $\mathcal{C}$ and $\mathcal{S}$, with or without decorations, range over subsets of $\mathcal{R}$. φ denotes a *fixed* acceptable programming system [Rog58,Rog67,Ric80,Ric81,Roy87]. φ_i denotes the partial computable function computed by program i in the φ-system. Note that in this paper all programs are interpreted with respect to the φ-system. We let Φ be an arbitrary Blum complexity measure [Blu67] associated with the acceptable programming system φ; many such measures exist for any acceptable programming system [Blu67]. Let $\varphi_{i,s}$ be defined as follows.

$$\varphi_{i,s}(x) = \begin{cases} \varphi_i(x), & \text{if } x < s \text{ and } \Phi_i(x) \leq s; \\ \uparrow, & \text{otherwise.} \end{cases}$$

For a given partial computable function η, we define MinProg(η) to denote $\min(\{i \mid \varphi_i = \eta\})$.

Let zeroext(η) denote a function defined as follows.

$$\mathrm{zeroext}(\eta)(x) = \begin{cases} \eta(x), & \text{if } x \in \mathrm{domain}(\eta); \\ 0, & \text{otherwise.} \end{cases}$$

2.1 Function Identification

We first describe inductive inference machines. We assume, without loss of generality, that the graph of a function is fed to a machine in canonical order. For any partial function η and $n \in N$ such that, for all $x < n$, $\eta(x)\downarrow$, we let $\eta[n]$ denote the finite initial segment $\{(x, \eta(x)) \mid x < n\}$. Clearly, $\eta[0]$ denotes the empty segment. We let Λ denote the empty segment. SEG denotes the set of all finite initial segments, $\{f[n] \mid f \in \mathcal{R} \wedge n \in N\}$. We let σ and τ, with or without decorations, range over SEG. Let $|\sigma|$ denote the length of σ. We often identify (partial) functions with their graphs. Thus for example, for $\sigma = f[n]$ and for $x < n$, $\sigma(x)$ denotes $f(x)$. A learning machine (also called an *inductive inference machine* (IIM)) [Gol67] is an algorithmic device that computes a mapping from SEG into $N \cup \{?\}$. Intuitively, "?" above denotes the case when the machine may not wish to make a conjecture. Although it is not necessary to consider learners that issue "?" for identification in the limit, it becomes useful when the number of mind changes a learner can make is bounded. In this paper, we assume, without loss of generality, that once an IIM has issued a conjecture on some initial segment of a function, it outputs a conjecture on all extensions of that initial segment. This is without loss of generality because a machine wishing to emit "?" after making a conjecture can instead be thought of as repeating its previous conjecture. We let $\mathbf{M}$, with or without decorations, range over learning machines. Since the set of all finite initial segments, SEG, can be coded onto N, we can view these machines as taking natural numbers as input and emitting natural numbers or ?'s as output. We say that $\mathbf{M}(f)$ converges to i (written: $\mathbf{M}(f)\downarrow = i$) iff $(\forall^\infty n)[\mathbf{M}(f[n]) = i]$; $\mathbf{M}(f)$ is undefined if no such i exists. The next definitions describe several criteria of function identification.

Definition 1 [Gol67,BB75,CS83] Let $a \in N \cup \{*\}$. Let $f \in \mathcal{R}$.

(a) $\mathbf{M}$ $\mathbf{Ex}^a$-*identifies* f (written: $f \in \mathbf{Ex}^a(\mathbf{M})$) just in case, there exists an i such that $\mathbf{M}(f)\downarrow = i$ and $\varphi_i =^a f$.

(b) $\mathbf{M}$ $\mathbf{Ex}^a$-*identifies* $\mathcal{S}$ iff $\mathbf{M}$ $\mathbf{Ex}^a$-identifies each $f \in \mathcal{S}$.

(c) $\mathbf{Ex}^a = \{\mathcal{S} \subseteq \mathcal{R} \mid (\exists \mathbf{M})[\mathcal{S} \subseteq \mathbf{Ex}^a(\mathbf{M})]\}$.

We often write $\mathbf{Ex}$ for $\mathbf{Ex}^0$.

By definition of convergence, only finitely many data points from a function f had been observed by an IIM $\mathbf{M}$ at the (unknown) point of convergence. Hence, some form of learning must take place in order for $\mathbf{M}$ to learn f. For this reason, hereafter the terms *identify*, *learn* and *infer* are used interchangeably.

Definition 2 [Bār74,CS83] Let $a \in N \cup \{*\}$. Let $f \in \mathcal{R}$.

(a) $\mathbf{M}$ $\mathbf{Bc}^a$-*identifies* f (written: $f \in \mathbf{Bc}^a(\mathbf{M})$) iff, for all but finitely many $n \in N$, $\varphi_{\mathbf{M}(f[n])} =^a f$.

(b) $\mathbf{M}$ $\mathbf{Bc}^a$-*identifies* $\mathcal{S}$ iff $\mathbf{M}$ $\mathbf{Bc}^a$-identifies each $f \in \mathcal{S}$.

(c) $\mathbf{Bc}^a = \{\mathcal{S} \subseteq \mathcal{R} \mid (\exists \mathbf{M})[\mathcal{S} \subseteq \mathbf{Bc}^a(\mathbf{M})]\}$.

We often write $\mathbf{Bc}$ for $\mathbf{Bc}^0$.

Some relationships between the above criteria are summarized in the following theorem.

Theorem 3 [CS83,BB75,Bār71]

$\mathbf{Ex}^0 \subset \mathbf{Ex}^1 \subset \cdots \subset \mathbf{Ex}^* \subset \mathbf{Bc} \subset \mathbf{Bc}^1 \subset \cdots \subset \mathbf{Bc}^* = 2^{\mathcal{R}}$.

Since $\mathcal{R} \in \mathbf{Bc}^*$, we often call a machine which $\mathbf{Bc}^*$-identifies $\mathcal{R}$ a general purpose learning machine.

We let $\mathbf{I}$ range over identification criteria defined above. There exists an r.e. sequence $\mathbf{M}_0, \mathbf{M}_1, \mathbf{M}_2, \ldots$, of inductive inference machines such that, for all criteria $\mathbf{I}$ of inference considered in this paper, one can show that [JORS99]:

for all $\mathcal{C} \in \mathbf{I}$, there exists an $i \in N$ such that $\mathcal{C} \subseteq \mathbf{I}(\mathbf{M}_i)$.

We assume $\mathbf{M}_0, \mathbf{M}_1, \mathbf{M}_2, \ldots$ to be one such sequence of machines.

3 General Purpose Machines and Their Mistakes

Unfortunately, the programs output by Harrington's machine become more and more degenerate, i.e. the finite set of anomalies in final programs output grows in size without bound. In fact the finite sets of anomalies cannot even be bounded by a computable function as the next theorem shows.

Theorem 4 *Suppose $\mathcal{R} \subseteq \mathbf{Bc}^*(\mathbf{M})$. Let g be a computable function. Then there exist infinitely many $f \in \mathcal{R}$ such that, for infinitely many n, $\varphi_{\mathbf{M}(f[n])} \neq^{g(n)} f$.*

PROOF. Suppose $\mathbf{M}$ and g are as given in the hypothesis. We will construct one f such that for infinitely many n, $\varphi_{\mathbf{M}(f[n])} \neq^{g(n)} f$. The construction can be easily modified to produce infinitely many distinct such f. By Kleene Recursion Theorem [Rog67], there exists an e such that φ_e may be defined as follows. Let $\varphi_e(0) = e$. Let $x_0 = 1$. Go to stage 0.

Stage s
1. Let $h = \text{zeroext}(\varphi_e[x_s])$.
2. Search for $n_s > x_s$ and a set S_s such that (a) $\varphi_{\mathbf{M}(h[n_s])}(y)\downarrow$, for all $y \in S_s$, (b) $\min(S_s) > n_s$, and (c) $\text{card}(S_s) > g(n_s)$.
3. If and when such n_s, S_s are found, let
   $\varphi_e(y) = 1 + \varphi_{\mathbf{M}(h[n_s])}(y)$, if $y \in S_s$.
   $\varphi_e(y) = h(y)$, if $y \leq \max(S_s)$ and $y \notin S_s$.
4. Let $x_{s+1} = 1 + \max(S_s)$.
   Go to stage $s + 1$.

End stage s

We first claim that step 2 succeeds in every stage. This is so since $\mathbf{M}$ $\mathbf{Bc}^*$-identifies all computable functions, and in particular h. Thus, there exist n_s and S_s such that (a) $\varphi_{\mathbf{M}(h[n_s])}(y)\downarrow$, for all $y \in S_s$, (b) $\min(S_s) > n_s$, and

(c) $\text{card}(S_s) > g(n_s)$. It follows that φ_e is total. Let $f = \varphi_e$. Now for all s, $\varphi_{\mathbf{M}(f[n_s])}(x) \neq f(x)$, for all $x \in S_s$. Thus, $\varphi_{\mathbf{M}(f[n_s])} \neq^{g(n_s)} f$. Theorem follows. $\square$

4 Predicting Near Future Values

Based on Theorem 4, it would be interesting to study how the anomalies of the programs outputted by a machine are distributed. For example, in real life one probably cares more about "near future errors" than "distant future errors". This leads us to the following definition.

Definition 5 Let $m, n \in N$ and $f \in \mathcal{R}$.

(a) $\mathbf{M}$ $\mathbf{Bc}^n_m$-*identifies* f (written: $f \in \mathbf{Bc}^n_m(\mathbf{M})$), iff, for all but finitely many x, $\text{card}(\{z < m \mid \varphi_{\mathbf{M}(f[x])}(x+z) \neq f(x+z)\}) \leq n$.

(b) $\mathbf{M}$ $\mathbf{Bc}^n_m$-*identifies* $\mathcal{C}$, if it $\mathbf{Bc}^n_m$-identifies each $f \in \mathcal{C}$.

(c) $\mathbf{Bc}^n_m = \{\mathcal{C} \mid \text{some } \mathbf{M}\ \mathbf{Bc}^n_m\text{-identifies } \mathcal{C}\}$.

Intuitively, one can view $\mathbf{Bc}^n_m$-identification of a function by a machine as follows. At any stage, the learning machine predicts the next m values. At all but finitely many stages, at least $m - n$ out of the m predictions are correct.

In this section we resolve the relationship between different $\mathbf{Bc}^n_m$-identification criteria.

Following four propositions follow directly from the definitions.

Proposition 6 *For $m \geq n$, $\mathbf{Bc}^n \subseteq \mathbf{Bc}^n_m$.*

Proposition 7 *Suppose $m \geq n$ and $k \in N$. Then $\mathbf{Bc}^n_m \subseteq \mathbf{Bc}^{n+k}_{m+k}$.*

Proposition 8 *Suppose $m \geq n \geq k$. Then $\mathbf{Bc}^k_m \subseteq \mathbf{Bc}^n_m$.*

Proposition 9 *Suppose $m \geq k \geq n$. Then $\mathbf{Bc}^n_m \subseteq \mathbf{Bc}^n_k$.*

Proposition 10 *For all $m > 0$, $\mathbf{Bc} = \mathbf{Bc}^0_m$.*

PROOF. Clearly, for $m \geq 1$, $\mathbf{Bc} \subseteq \mathbf{Bc}^0_m \subseteq \mathbf{Bc}^0_1$. $\mathbf{NV}''$ defined by Podnieks [Pod74] is identical to $\mathbf{Bc}^0_1$. The proposition follows from $\mathbf{NV}'' = \mathbf{Bc}$ [Pod74,CS83]. $\square$

The following theorem shows some advantages of having to predict fewer correct values in the near future.

Theorem 11 *Suppose $m' > m$. Then,* $\mathbf{Bc}^1_m - \mathbf{Bc}^{m'-m}_{m'} \neq \emptyset$.

PROOF. Let

$$Z_k = \{x \mid k \cdot m' < x < k \cdot m' + m\},$$

$$E_k = \{k \cdot m'\} \cup \{x \mid k \cdot m' + m \leq x < (k+1) \cdot m'\},$$

and

$$U_k = Z_k \cup E_k = \{x \mid k \cdot m' \leq x < (k+1) \cdot m'\}.$$

We now consider the following two properties defined on total functions.

(PropA) f satisfies PropA iff, for all k, for all $x \in Z_k$, $f(x) = 0$.
(PropB) f satisfies PropB iff, for all k, for all $x \in E_k$, $f(x) = f(k \cdot m')$.

Let $\mathcal{C} = \{f \in \mathcal{R} \mid f$ satisfies PropA and PropB$\}$.

The above class is easily seen to be in $\mathbf{Bc}^1_m$. We now show that $\mathcal{C} \notin \mathbf{Bc}^{m'-m}_{m'}$.

Suppose by way of contradiction that $\mathbf{M}$ $\mathbf{Bc}^{m'-m}_{m'}$-identifies $\mathcal{C}$. Let $\sigma_0 = \Lambda$. In the following construction, in stage s we will define σ_{s+1}. Domain of σ_s will be $\bigcup_{k<s} U_k$. Also, $\lim_{s\to\infty} \sigma_s(x)\downarrow$. Suppose f is defined as follows:

$$f(x) = \lim_{s\to\infty} \sigma_s(x)$$

Then, f will be in $\mathcal{C}$, and $\mathbf{M}$ does not $\mathbf{Bc}^{m'-m}_{m'}$-identifies f.

For all k, let $\mathrm{err}^k_0 = 0$. Intuitively, at the beginning of stage s, for $k < s$, err^k_s denotes the number of inputs in E_k on which $\mathbf{M}(\sigma_s[k \cdot m'])$ has been currently diagonalized against. We will have $\mathrm{err}^k_s = 0$, for $k \geq s$.

Go to stage 0.

Stage s
1. For $k \leq s$, let $\mathrm{Conv}^k_s = \mathrm{card}(\{x \in E_k \mid \varphi_{\mathbf{M}(\sigma_s[k\cdot m']),s}(x)\downarrow\})$.
2. Let $\mathrm{Cand}_s = \{k \leq s \mid \mathrm{Conv}^k_s > \max(\{err^{k'}_s \mid k \leq k' \leq s\})\}$.
   If $\mathrm{Cand}_s = \emptyset$, then let $C_s = s$. Otherwise, let C_s be the minimum element in Cand_s.
3. For $k < C_s$, let $\mathrm{err}^k_{s+1} = \mathrm{err}^k_s$.
   Let $\mathrm{err}^{C_s}_{s+1} = \mathrm{Conv}^{C_s}_s$.
   For $k > C_s$, let $\mathrm{err}^k_{s+1} = 0$.
4. For $x \in \bigcup_{k<C_s} U_k$, let $\sigma_{s+1}(x) = \sigma_s(x)$.
   For $x \in E_{C_s}$, let $\sigma_{s+1}(x) = 1 + \max(\{\varphi_{\mathbf{M}(\sigma_s[C_s\cdot m'])}(x) \mid x \in E_{C_s} \wedge \varphi_{\mathbf{M}(\sigma_s[C_s\cdot m']),s}(x)\downarrow\})$.
   For $x \in Z_{C_s} \cup \bigcup_{C_s<k<s+1} U_k$, let $\sigma_{s+1}(x) = 0$.
5. Go to stage $s+1$.

End stage s

Claim 12 *For all s, the following three properties hold.*

(a) for all $x \in \bigcup_{k<C_s} U_k$, $\sigma_{s+1}(x) = \sigma_s(x)$.

(b) for all $k < C_s$, $\mathrm{err}^k_{s+1} = \mathrm{err}^k_s$, and

(c) either $C_s = s$, or $\mathrm{err}^{C_s}_{s+1} > \max(\{\mathrm{err}^k_s \mid C_s \leq k \leq s\})$.

Proof. Follows directly from construction. □ (Claim 12)

Claim 13 $\lim_{s\to\infty} \mathrm{err}^k_s\downarrow$ *for every k. Thus,* $\liminf C_s$ *goes to infinity, and* $\lim_{s\to\infty} \sigma_s(x)\downarrow$.

Proof.

We show the claim by induction on k. Suppose, for $k' < k$, $\lim_{s\to\infty} \mathrm{err}^{k'}_s\downarrow$. Let $t > k$ be a stage such that, for all $k' < k$, for all $s > t$, $\mathrm{err}^{k'}_s = \mathrm{err}^{k'}_t$. Thus, by Claim 12(b and c), for all $s > t$, $C_s \geq k$. Hence, by Claim 12(b and c), for all $s > t$, $\mathrm{err}^k_s \leq \mathrm{err}^k_{s+1}$. Since err^k_s is bounded by $m' - m + 1$, $\lim_{s\to\infty} \mathrm{err}^k_s\downarrow$. Thus, using Claim 12, it follows that $\liminf C_s$ goes to infinity, and thus $\lim_{s\to\infty} \sigma_s(x)\downarrow$. □ (Claim 13)

Let

$$f(x) = \lim_{s\to\infty} \sigma_s(x)$$

We claim that $f \in \mathcal{C}$ and $\mathbf{M}$ does not $\mathbf{Bc}^{m'-m}_{m'}$-identify f. Note that f clearly satisfies PropA and PropB. We thus just need to show that $f \in \mathcal{R}$ and $\mathbf{M}$ does not $\mathbf{Bc}^{m'-m}_{m'}$-identify f.

Let err^k denote $\lim_{s\to\infty} \mathrm{err}^k_s$. Let $r \leq m' - m + 1$, be the largest value such that $\mathrm{err}^k = r$, for infinitely many k.

Let t_1, t_2 be large enough so that, $t_2 > t_1$, and

(i) for all $k \geq t_1$, $\mathrm{err}^k \leq r$;

(ii) for all $k \leq t_1$, for all $s \geq t_2$, $\mathrm{err}^k_s = \mathrm{err}^k_{t_2}$.

Thus, in particular for all $s > t_2$, $\mathrm{err}^{C_s}_s \leq r$.

Claim 14 *For all $s > t_2$, if $err^{C_s}_s = r$, then for all $s' > s$, $C_{s'} > C_s$,*

Proof. Suppose by way of contradiction that $s' > s$ is the first stage in which $C_{s'} \leq C_s$. Then $err^{C_{s'}}_{s'} > err^{C_s}_s$, by the condition for selection of $C_{s'}$ in step 2 of the construction. This in turn implies that $\mathrm{err}^{C_{s'}}_{s'} > r$. A contradiction to the choice of t_2. □ (Claim 14)

It follows from Claims 12 and 14 that, if $s > k_2$ and $\mathrm{err}_s^{C_s} = r$, then $\sigma_{s+1}[(C_s + 1) \cdot m'] \subseteq f$. Moreover, since $\mathrm{err}_s^{C_s} = r$, for infinitely many s, it follows that

$\bigcup_{\{s \mid s>t_2 \ \wedge\ \mathrm{err}_s^{C_s}=r\}} \sigma_{s+1}[(C_s + 1) \cdot m'] = f.$

Since the left hand side of above equation is computable, it follows that f is computable.

Furthermore, for all $s > t_2$, for all $k > t_1$, $\mathrm{Conv}_s^k \leq r$ (otherwise $\mathrm{err}_s^{C_s}$ would be at least $r + 1$).

Thus, for all $C_s > t_1$, such that $\mathrm{err}_s^{C_s} = r$, we have $\varphi_{\mathbf{M}(f[C_s \cdot m'])}(x) \neq f(x)$, for all $x \in E_{C_s}$. It follows that $\mathbf{M}$ does not $\mathbf{Bc}_{m'}^{m'-m}$-identify f. Theorem follows. $\Box$(Theorem 11)

The next theorem shows some advantages of being allowed to predict more wrong values in the near future.

Theorem 15 *For all $n \in N$, $\mathbf{Bc}^{n+1} - \mathbf{Bc}_{n+1}^{n} \neq \emptyset$.*

PROOF. Let $Z_f = \{x \mid f((n + 2) \cdot x) \neq 0\}$.

Let $\mathcal{C} = \{f \mid [\mathrm{card}(Z_f) = \infty \wedge (\forall^\infty x \in Z_f)[\varphi_{f((n+2)\cdot x)} =^{n+1} f]] \ \vee\ [0 < \mathrm{card}(Z_f) < \infty \wedge\ \varphi_{f((n+2)\cdot \max(Z_f))} =^{n+1} f]\}$.

It is easy to verify that $\mathcal{C} \in \mathbf{Bc}^{n+1}$. We now show that $\mathcal{C} \notin \mathbf{Bc}_{n+1}^{n}$. Suppose by way of contradiction, $\mathbf{M}$ $\mathbf{Bc}_{n+1}^{n}$-identifies $\mathcal{C}$. Then, by operator recursion theorem [Cas74], there exists a computable, 1–1, increasing p such that $p(0) > 0$, and $\varphi_{p(i)}$ may be described in stages as follows.

Below, let $\varphi_{p(y)}^s$ denote $\varphi_{p(y)}$ defined before stage s. Let $X_k = \{x \mid (n+2) \cdot k < x < (n+2) \cdot (k+1)\}$. Intuitively, for the diagonalizing function f constructed, for infinitely many k, for all $x \in X_k$, $\varphi_{\mathbf{M}(f[(n+2)\cdot k+1])}(x) \neq f(x)$.

We will define variables q_s^i (for $i \leq n + 2$), σ_s, and E_s^i (for $i \leq n + 1$) in the construction. Intuitively, for the diagonalizing function f which we will construct, think of q_s^i (for $i \leq n+1$) as elements of Z_f. σ_s is an approximate initial segment of f at the beginning of stage s. The domain of σ_s will be $\{x \mid x < (n + 2) \cdot (1 + q_s^{n+2})\}$. E_s^i denotes a set of k's such that $q_s^i < k \leq q_s^{i+1}$ and $\varphi_{\mathbf{M}(\sigma_s[(n+2)\cdot k+1])}$ makes at least $n + 1 - i$ convergent errors on inputs from X_k (with respect to σ_s).

For $i \leq n+2$, let $q_0^i = i$. Initial segment σ_0, with domain $\{x \mid x < (n+2) \cdot (q_0^{n+2}+1)\}$ is defined as follows.

$$\sigma_0(x) = \begin{cases} p(q_0^i), & \text{if } x = q_0^i \cdot (n+2), \text{ for some } i < n+2; \\ 0, & \text{if } x < (n+2) \cdot (q_0^{n+2}+1) \text{ and } x \text{ is not of} \\ & \text{form } q_0^i \cdot (n+2), \text{ for any } i < n+2. \end{cases}$$

For $i \leq n+1$, let $\varphi^0_{p(q_s^i)} = \sigma_s[(n+2) \cdot (1 + q_s^{i+1})]$.

For $i < n+1$, let $E_0^i = \emptyset$, and let $E_0^{n+1} = \{q_s^{n+2}\}$. In stage s, we will define σ_{s+1} and q_{s+1}^i, for $i \leq n+2$, and correspondingly define E_{s+1}^i, for $i \leq n+1$. This will be done is such a way, so that (a) $(\forall x)(\forall^\infty s)[\sigma_{s+1}(x) = \sigma_s(x)]$, and (b) f defined as $f(x) = \lim_{s\to\infty} \sigma_s(x)$, is the diagonalizing function.

For all s, we will satisfy the following invariants:

(A) For $i \leq n$, $q_s^i < q_s^{i+1}$.

(B) For all $i \leq n+1$, $\varphi^s_{p(q_s^i)} = \sigma_s[(n+2) \cdot (q_s^{i+1}+1)]$.

(C) For all $i \leq n+1$, for all $k \in E_s^i$, $\mathrm{card}(\{x \in X_k \mid \varphi^s_{\mathbf{M}(\sigma_s[(n+2)\cdot k+1])}(x)\downarrow \neq \sigma_s(x)\}) \geq n+1-i$.

(D) For all $i \leq n+1$, $E_s^i \subseteq \{x \mid q_s^i < k \leq q_s^{i+1}\}$.

Go to stage 0.

Stage s

1. Let $\mathrm{Cand}_s = \{i < n+1 \mid (\exists k \in E_s^{i+1})[\mathrm{card}(\{x \in X_k \mid \varphi_{\mathbf{M}(\sigma_s[k\cdot(n+2)+1]),s}(x)\downarrow\}) \geq n+1-i]\}$.
2. If Cand_s is empty, then let $C_s = n+1$. Else, let $C_s = \min(\mathrm{Cand}_s)$.
3. If $C_s < n+1$, then let $k_s \in E_s^{C_s+1}$ be such that $\mathrm{card}(\{x \in X_{k_s} \mid \varphi_{\mathbf{M}(\sigma_s[k_s\cdot(n+2)+1]),s}(x)\downarrow\}) \geq n+1-i$.
   If $C_s < n+1$ and there exists an $x \in X_{k_s}$ such that $\sigma_s(x) = \varphi_{\mathbf{M}(\sigma_s[k_s\cdot(n+2)+1]),s}(y_s)\downarrow$, then let y_s be one such x. Else, let $y_s = \uparrow$.
4. For $i \leq C_s$, let $q_{s+1}^i = q_s^i$.
   For $C_s < i \leq n+2$, let $q_{s+1}^i = q_s^{n+2} + i - C_s$.

5. Let

$$\sigma_{s+1}(x) = \begin{cases} \sigma_s(x), & \text{if } x < (n+2)\cdot(1+q_s^{n+2}) \\ & \text{and } x \neq y_s; \\ \varphi_{\mathbf{M}(\sigma_s[k_s\cdot(n+2)+1]),s}(y_s)+1, & \text{if } x = y_s; \\ p(q_{s+1}^i), & \text{if } (n+2)\cdot(1+q_s^{n+2}) \leq x \\ & \text{and } x < (n+2)\cdot(1+q_{s+1}^{n+2}) \\ & \text{and } x = q_{s+1}^i \cdot (n+2), \\ & \text{for some } i < n+2; \\ 0, & \text{if } (n+2)\cdot(1+q_s^{n+2}) \leq x \\ & \text{and } x < (n+2)\cdot(1+q_{s+1}^{n+2}) \\ & \text{and } x \text{ is not of form} \\ & q_{s+1}^i \cdot (n+2), \\ & \text{for any } i < n+2. \end{cases}$$

6. For $i < C_s$, let $E_{s+1}^i = E_s^i$.
   If $C_s < n+1$, then let $E_{s+1}^{C_s} = E_s^{C_s} \cup \{k_s\}$.
   For $C_s < i < n+1$, let $E_{s+1}^i = \emptyset$.
   Let $E_{s+1}^{n+1} = \{k \mid q_{s+1}^{n+1} < k \leq q_{s+1}^{n+2}\}$.
7. For $i \leq n+1$, let $\varphi_{p(q_{s+1}^i)}^{s+1} = \sigma_{s+1}[(n+2)\cdot(q_{s+1}^{i+1}+1)]$.
8. For $C_s < i \leq n+1$, let $\varphi_{p(q_s^i)}$ follow $\varphi_{p(q_{s+1}^{C_s})}$ on inputs $\geq (n+2)\cdot(q_s^{i+1}+1)$.
   (* Note that above implies that $\varphi_{p(q_s^i)} =^1 \varphi_{p(q_{s+1}^{C_s})}$, since $\varphi_{p(q_s^i)}$ and $\varphi_{p(q_{s+1}^{C_s})}$ may differ only on y_s. *)
9. Go to stage $s+1$.

End stage s

It is easy to verify that invariants (A) to (D) are satisfied. Also, note that step 7 is consistent since, for $i \leq C_s$, $\sigma_s[(n+2)\cdot(q_s^{i+1}+1)] \subseteq \sigma_{s+1}[(n+2)\cdot(q_{s+1}^{i+1}+1)]$ (by invariant (D), if $C_s < n+1$, then $k_s > q_s^{C_s+1}$ and thus $y_s > (n+2)\cdot(q_s^{C_s+1}+1)$).

Claim 16 *For each s, the following are satisfied.*

(a) For $i \leq C_s$, $q_{s+1}^i = q_s^i$.

(b) For $C_s < i \leq n+2$, $q_{s+1}^i > q_s^{n+2}$.

(c) $q_s^i \leq q_{s+1}^i$.

(d) For $i < j$, if $q_s^i < q_{s'}^j$, then $s \leq s'$ or $q_s^i = q_{s'}^i$.

(e) $\sigma_s[(n+2)\cdot(q_s^{C_s+1}+1)] \subseteq \sigma_{s+1}[(n+2)\cdot(q_s^{C_s+1}+1)]$.

(f) For all $C_s < i \leq n+1$, $\varphi_{p(q_s^i)} =^1 \varphi_{p(q_s^{C_s})}$.

PROOF. Parts (a) to (e) follow by induction on stages. For (f) see note in step 8. □ (Claim 16)

Claim 17 *For all s and for all $i < n+1$,* $\mathrm{card}(\{x \mid \varphi_{p(q_s^i)}(x)\downarrow \neq \varphi_{p(q_s^{i+1})}(x)\}) \leq 1$.

PROOF. If for all $s' \geq s$, $q_{s'}^{i+1} = q_s^{i+1}$, then clearly, $\varphi_{p(q_s^i)} = \varphi_{p(q_s^i)}^s = \sigma_s[(n+2)\cdot(q_s^{i+1}+1)] \subseteq \varphi_{p(q_s^{i+1})}^s \subseteq \varphi_{p(q_s^{i+1})}$.

So suppose $s' > s$ is the least number such that $q_{s'}^{i+1} \neq q_s^{i+1}$. Then, by construction of $\sigma_{s'}$ and steps 7 and 8 in stage $s'-1$, the only possible member of $\{x \mid \varphi_{p(q_s^i)}(x)\downarrow \neq \varphi_{p(q_s^{i+1})}(x)\}$ is $y_{s'-1}$. Claim follows. □ (Claim 17)

By inductively applying above claim we get,

Corollary 18 *For all s and for all $i < j \leq n+1$,* $\mathrm{card}(\{x \mid \varphi_{p(q_s^i)}(x)\downarrow \neq \varphi_{p(q_s^j)}(x)\}) \leq j-i$.

Let $C \leq n+1$, be the least value such that $C_s = C$ for infinitely many s. Thus, for $i \leq C$, $\lim_{s\to\infty} q_s^i\downarrow$, and for $C < i \leq n+1$, $\lim_{s\to\infty} q_s^i\uparrow$. For $i \leq C$, let $q^i = \lim_{s\to\infty} q_s^i$. Claim 16(c) and invariant (B) thus imply that $\varphi_{p(q^C)}$ is total. Let $f = \varphi_{p(q^C)}$. Note that for all $x \in Z_f$, $f(x) = p(x)$ and $x = q_s^j$, for some $s \in N$ and $j \leq n+1$ (by construction).

Claim 19 $f \in \mathcal{C}$.

PROOF. Suppose $q_s^j > q^C$, and $j \leq n+1$. Then, we must have $j > C$, and thus by Claim 16 (d), $q_s^C = q^C$. Thus, by Corollary 18 we have that, for all $q_s^j > q^C$, $\varphi_{p(q^C)} =^{j-C} \varphi_{p(q_s^j)}$. Thus $f \in \mathcal{C}$. □ (Claim 19)

Let s be such that, for all $s' \geq s$, $q_s^C = q^C$. Let $E^C = \bigcup_{s'\geq s} E_s^C$. Note that E^C is infinite.

Claim 20 *For all $k \in E^C$, for all $x \in X_k$, $\varphi_{\mathbf{M}(f[(n+2)\cdot k+1])}(x) \neq f(x)$.*

PROOF. For all $k \in E^C$, $\varphi_{\mathbf{M}(f[(n+2)\cdot k+1])}$ converges on at most $n+1-C$ elements in X_k (otherwise step 3 in the construction would make $C_{s'} < C$, for some $s' > s$). Moreover, due to invariant (C), for all $k \in E^C$, $\varphi_{\mathbf{M}(f[(n+2)\cdot k+1)]}$ makes at least $n+1-C$ convergent errors in X_k. It follows that, for $k \in E^C$, for each $x \in X_k$, $\varphi_{\mathbf{M}(f[(n+2)\cdot k+1])}(x) \neq f(x)$. □ (Claim 20)

Since, E^C is infinite it follows from above claim that $\mathbf{M}$ does not $\mathbf{Bc}_{n+1}^n$-identify f. Theorem follows. □(Theorem 15)

As a corollary to the above theorem, if one looks at the errors committed by a general purpose machine on the next n inputs, then for infinitely many functions, at infinitely many positions, the machine commits n errors in predicting the next n inputs. Hence, there is an ostensibly unpleasant cost to general purpose learning. However, as we shall see, this can be assuaged at least in some interesting respects (Theorems 28 and 30 in Section 5 below).

Corollary 21 *Suppose $m > n$ and $m' > n'$. If $n > n'$, then $\mathbf{Bc}^n_m - \mathbf{Bc}^{n'}_{m'} \neq \emptyset$.*

PROOF. Theorem 15 shows that $\mathbf{Bc}^{n'+1} - \mathbf{Bc}^{n'}_{n'+1} \neq \emptyset$. Now since $\mathbf{Bc}^{n'+1} \subseteq \mathbf{Bc}^n \subseteq \mathbf{Bc}^n_m$ (latter inclusion by Proposition 6) and $\mathbf{Bc}^{n'}_{m'} \subseteq \mathbf{Bc}^{n'}_{n'+1}$ (by Proposition 9), Corollary follows. $\square$

Corollary 22 *Suppose $m' > n'$ and $m > n > 0$. If $m' - n' > m - n$, then $\mathbf{Bc}^n_m - \mathbf{Bc}^{n'}_{m'} \neq \emptyset$.*

PROOF. If $n' < n$, then corollary follows from Corollary 21. So suppose $n \leq n'$. Theorem 11 shows that $\mathbf{Bc}^1_{m-n+1} - \mathbf{Bc}^{m'-m+n-1}_{m'} \neq \emptyset$ (note that $m' > m - n + 1$). Now, $\mathbf{Bc}^{1+n-1}_m \supseteq \mathbf{Bc}^1_{m-n+1}$ (by Proposition 7), and $\mathbf{Bc}^{n'}_{m'} \subseteq \mathbf{Bc}^{m'-m+n-1}_{m'}$ (since $n' \leq m' - m + n - 1$, and by Proposition 8). Corollary follows. $\square$

The following corollary resolves all relationships among the $\mathbf{Bc}^n_m$-criteria.

Corollary 23 *Suppose $m > n$ and $m' > n'$. Then: $\mathbf{Bc}^n_m \subseteq \mathbf{Bc}^{n'}_{m'}$ iff $[n = 0$ or $[n' \geq n$ and $m' - n' \leq m - n]]$.*

PROOF. If $n > n'$ then Corollary 21 shows that $\mathbf{Bc}^n_m - \mathbf{Bc}^{n'}_{m'} \neq \emptyset$. If $m' - n' > m - n$ and $n > 0$, then Corollary 22 shows that $\mathbf{Bc}^n_m - \mathbf{Bc}^{n'}_{m'} \neq \emptyset$.

If $n = 0$, then $\mathbf{Bc}^n_m = \mathbf{Bc} \subseteq \mathbf{Bc}^{n'}_{m'}$.

If $n' \geq n$ and $m' - n' \leq m - n$, then $\mathbf{Bc}^n_m \subseteq \mathbf{Bc}^{n'}_{m+n'-n}$ (by Proposition 7) and $\mathbf{Bc}^{n'}_{m+n'-n} \subseteq \mathbf{Bc}^{n'}_{m'}$ (by Proposition 9). Corollary follows. $\square$

Corollary 24 *For all $m > n$, $\mathcal{R} \notin \mathbf{Bc}^n_m$.*

Thus, no general purpose learning machine can guarantee that anomalies are not concentrated in the near future.

5 Desirable Properties Achievable By General Purpose Learners

Since the errors committed by programs output by a general purpose learner can be arbitrarily bad, we look at how this may be assuaged for suitable general purpose learners, and we also determine some additional nice properties a general purpose learner can satisfy.

On can think of a program for a computable function as a *predictive explanation* for the function's I/O behavior [BB75,CS83]. Popper's Refutability Principle [Pop68] essentially says that explanations with mistakes should be refutable. As pointed out in [CS83] (see also [CJNM94]), an erroneous predictive explanation (program) for a computable function satisfies Popper's Principle if it computes a total function. [2] The following theorem says that one can construct a general purpose learner which, on computable function input, almost always outputs programs for total functions; hence, it almost always outputs predictive explanations which satisfy Popper's Principle.

Theorem 25 *There exists a machine* $\mathbf{M}$*, such that, for all* $f \in \mathcal{R}$*, (i)* $\mathbf{M}$ $\mathbf{Bc}^*$*-identifies* f*, and (ii)* $(\forall^\infty n)[\varphi_{\mathbf{M}(f[n])} \in \mathcal{R}]$.

PROOF. Define $\mathbf{M}$ as follows. $\mathbf{M}$ on $f[n]$, outputs a program p_n such that φ_{p_n} may be defined as follows.

$\varphi_{p_n}(x)$
1. Let $S_x = \{p \leq n \mid f[n] \subseteq \varphi_{p,x}\}$.
2. If $S_x = \emptyset$, then let $\varphi_{p_n}(x) = 0$.
3. Else let $p = \min(S_x)$.
4. Dovetail steps 5 and 6 until one of them succeeds. If step 5 succeeds, before step 6 (if ever) then go to step 7. If step 6 succeeds, before step 5 (if ever) then go to step 8.
5. Search for s such that $\varphi_{p,s}(x)\downarrow$.
6. Search for $p' < p$, and s such that, $(\forall y < n)[\varphi_{p',s}(y) = f(y)]$ and $\varphi_{p',s}(x)\downarrow$.
7. Let $\varphi_{p_n}(x) = \varphi_p(x)$, and halt.
8. Let $\varphi_{p_n}(x) = \varphi_{p'}(x)$.

End

We claim that above $\mathbf{M}$ witnesses the theorem. Suppose $f \in \mathcal{R}$. Let q be the least program for f. Let $m > q$ be large enough so that, for all $q' < q$, there exists a $y < m$ such that $\varphi_{q'}(y) \neq f(y)$. Let $p_n = \mathbf{M}(f[n])$, and consider the definition of φ_{p_n} above.

Claim 26 $(\forall n > m)[\varphi_{p_n} =^* f]$. *Thus,* $\mathbf{M}$ $\mathbf{Bc}^*$*-identifies* f.

PROOF. Note that for large enough x, $\min(S_x) = q$. Also, by definition of m, search in step 6 cannot succeed for any $p' < p = q$. Also, for $p = q$, there exists an s such that search in step 5 succeeds. Thus, for large enough x, $\varphi_{p_n}(x) = \varphi_q(x) = f(x)$. □ (Claim 26)

Claim 27 $(\forall n > m)[\varphi_{p_n} \in \mathcal{R}]$.

[2] Then the halting problem [Rog67] does not stand in the way of algorithmically locating the mistakes.

PROOF. Note that, for $n > m$, $\min(S_x) \geq q$. Thus, for all x, step 6 of $\varphi_{p_n}(x)$ would eventually succeed since $p' = q$ and large enough s satisfy the requirement for success of step 6. Thus, $\varphi_{p_n}(x)$ would be defined at step 8 (if not earlier defined due to step 2 or 7). $\square$ (Claim 27)

Theorem follows from above claims. $\square$(Theorem 25)

The following shows that even though a general purpose machine may be locally bad for infinitely many positions, one can ensure that these bad positions have low density.

Theorem 28 *For all n, there exists a machine* $\mathbf{M}$ *such that,*

(a) $\mathbf{M}$ $\mathbf{Bc}^*$*-identifies* $\mathcal{R}$*, and*

(b) for all $f \in \mathcal{R}$, $\lim_{x\to\infty} \frac{\mathrm{card}(\{k \mid k \leq x \ \wedge\ (\forall z \leq n)[\varphi_{\mathbf{M}(f[k])}(k+z) = f(k+z)]\})}{x+1} = 1$.

PROOF.

Suppose $\mathbf{M}'$ $\mathbf{Bc}^*$-identifies $\mathcal{R}$. Suppose h is a monotonic non-decreasing computable function such that $\lim_{x\to\infty} \frac{h(x)}{x} = 0$. Let $\mathbf{M}$ be defined as follows.

$\mathbf{M}(f[m])$ outputs a program p_m defined as follows.

$\varphi_{p_m}(y)$
1. If $y < m$, then $\varphi_{p_m}(y) = f(y)$.
2. If $y > m + n$, then $\varphi_{p_m}(y) = \varphi_{\mathbf{M}'(f[m])}(y)$.
3. If $m \leq y \leq m + n$, then search for a $p < h(m)$ such that $f[m] \subseteq \varphi_p$, and $\varphi_p(y)\downarrow$. If and when such a p is found, let $\varphi_{p_m}(y) = \varphi_p(y)$.

End

Due to step 2, $\varphi_{\mathbf{M}(f[m])} =^* \varphi_{\mathbf{M}'(f[m])}$. Thus, $\mathbf{M}$ $\mathbf{Bc}^*$-identifies $\mathcal{R}$. Now fix $f \in \mathcal{R}$. Fix x. Consider $m \leq x$ such that $\varphi_{\mathbf{M}(f[m])}(m+z) \neq f(m+z)$, for some $z \leq n$. For these m one of the following two conditions must hold.

Case 1: For all $p < h(m)$, $[f[m] \not\subseteq \varphi_p$ or $\varphi_p(m+z)\uparrow$ for some $z \leq n]$.

Case 2: There exists a $p < h(m)$ such that $f[m] \subseteq \varphi_p$ and $\varphi_p(m+z)\downarrow \neq f(m+z)$, for some $z \leq n$.

Case 1 can hold for only for $h(m) \leq \mathrm{MinProg}(f)$. Each $p < h(x)$, can result in Case 2 for at most $n+1$ different m (since if w is the least number such that $\varphi_p(w) \neq f(w)$, then for Case 2 to happen, $w - n \leq m \leq w$). Thus, Case 2 can happen for a total of at most $h(x) * (n+1)$ different $m \leq x$. Let c be such that $h(c) > \mathrm{MinProg}(f)$. Thus, Case 1 or Case 2 can happen for at most $c + h(x) * (n+1)$ different m. Now, $\lim_{x\to\infty} \frac{c+h(x)*(n+1)}{x+1} = 0$ (since c and n are constants and $\lim_{x\to\infty} h(x)/x = 0$). Since f was arbitrary computable function, theorem follows. $\square$

Since general purpose learners are always quite erroneous (of course the density of erroneous, near future intervals can be made small), it is interesting to consider which classes a general purpose learner may additionally identify in a better or stricter sense.

Definition 29 [Min76,BB75,CJNM94]

(a) $\mathbf{M}$ is said to be *reliable* iff, for all f such that $\mathbf{M}(f)\downarrow$, $\mathbf{M}$ $\mathbf{Ex}$-identifies f.

(b) $\mathbf{M}$ is said to *reliably* $\mathbf{Ex}$*-identify* $\mathcal{C}$, iff $\mathbf{M}$ is reliable and $\mathbf{M}$ $\mathbf{Ex}$-identifies $\mathcal{C}$.

(c) $\mathbf{RelEx} = \{\mathcal{C} \mid$ some machine reliably $\mathbf{Ex}$-identifies $\mathcal{C}\}$.

Intuitively, reliable machines do not deceive us by converging falsely. As noted above, r.e. classes of computable functions (such as the class of polynomial functions and the class of primitive recursive functions) as well as the class of total run time functions can be reliably $\mathbf{Ex}$-identified.

The following theorem shows that for any class $\mathcal{S}$ in $\mathbf{RelEx}$, one can create a general purpose learning machine which $\mathbf{Ex}$-identifies $\mathcal{S}$!

Theorem 30 *Suppose* $\mathcal{S} \in \mathbf{RelEx}$. *Then there exists an* $\mathbf{M}$ *such that* $\mathbf{M}$ $\mathbf{Bc}^*$*-identifies* $\mathcal{R}$ *and* $\mathbf{Ex}$*-identifies* $\mathcal{S}$.

PROOF.

Suppose $\mathbf{M}_H$ is a machine which $\mathbf{Bc}^*$-identifies $\mathcal{R}$. Below P ranges over finite sets of programs. Let Prog be a recursive function such that $\varphi_{\mathrm{Prog}(P,f[n])}$ may be defined as follows.

$\varphi_{\mathrm{Prog}(P,f[n])}(x)$
1. If there exists a $y \leq x$ and $i, j \in P$ such that
   $\Phi_i(y) \leq x$, $\Phi_j(y) \leq x$ and $\varphi_i(y) \neq \varphi_j(y)$.
   Then let $\varphi_{\mathrm{Prog}(P,f[n])}(x) = \varphi_{\mathbf{M}_H(f[n])}(x)$.
2. Else, search for an $i \in P$ such that $\varphi_i(x)\downarrow$. If and when such an i is found,
   let $\varphi_{\mathrm{Prog}(P,f[n])}(x) = \varphi_i(x)$.

End

Claim 31 *For all* $f \in \mathcal{R}$, *for all* n, *for all* P,

$[(\exists p \in P)[\varphi_p = f] \wedge (\forall p \in P)[\varphi_p \subseteq f]] \Rightarrow [\varphi_{\mathrm{Prog}(P, f[n])} = f]$.

PROOF. Fix $f \in \mathcal{R}$, and $n \in N$. Suppose, $(\exists p \in P)[\varphi_p = f] \wedge (\forall p \in P)[\varphi_p \subseteq f]$. Fix any input x. Since $(\forall p \in P)[\varphi_p \subseteq f]$, 'If' clause in step 1 of $\varphi_{\mathrm{Prog}(P,f[n])}(x)$ fails. Also, there exists a $p' \in P$, such that $\varphi_{p'}(x)\downarrow$. It follows that search in step 2 of $\varphi_{\mathrm{Prog}(P,f[n])}(x)$, succeeds. Thus, $\varphi_{\mathrm{Prog}(P,f[n])}(x)\downarrow$. $\varphi_{\mathrm{Prog}(P,f[n])}(x) = f(x)$ now follows from the hypothesis that $(\forall p \in P)[\varphi_p \subseteq f]$. $\square$ (Claim 31)

Claim 32 *For all $f \in \mathcal{R}$, for all but finitely many n, for all P,*

$(\exists p \in P)[\varphi_p = f] \Rightarrow [\varphi_{\mathrm{Prog}(P,f[n])} =^* f]$.

PROOF.

Fix $f \in \mathcal{R}$. Let n_0 be such that, for all $n \geq n_0$, $\varphi_{\mathbf{M}_H(f[n])} =^* f$.

Suppose there exists a $p \in P$ such that $\varphi_p = f$. If, $(\forall p \in P)[\varphi_p \subseteq f]$, then by Claim 31 $\varphi_{\mathrm{Prog}(P,f[n])} = f$.

If, $(\exists p \in P)[\varphi_p \not\subseteq f]$, then using $(\exists p \in P)[\varphi_p = f]$, it follows that, there exist $i, j \in P$, $x_0, y_0 \in N$, such that $y_0 \leq x_0$, and $\Phi_i(y_0) \leq x_0$, $\Phi_j(y_0) \leq x_0$ and $\varphi_i(y_0) \neq \varphi_j(y_0)$. Thus, for all $x \geq x_0$, $\varphi_{\mathrm{Prog}(P,f[n])}(x) = \varphi_{\mathbf{M}_H(f[n])}(x)$. It follows that, for $n \geq n_0$, $\varphi_{\mathrm{Prog}(P,f[n])} =^* \varphi_{\mathbf{M}_H(f[n])} =^* f$. Claim follows. $\square$ (Claim 32)

We now show the theorem using above claims. Suppose $\mathbf{M}'$ **RelEx**-identifies $\mathcal{S}$. Without loss of generality assume $\mathcal{S} = \mathbf{Ex}(\mathbf{M}')$, and for all n, $\mathbf{M}'(f[n]) \leq n$. Let $\mathrm{SB}(f[n]) = \max(\{\mathbf{M}'(f[x]), \mathrm{card}(\{m < n-1 \mid \mathbf{M}'(f[m]) \neq \mathbf{M}'(f[m+1])\})\})$. Note that if $\mathbf{M}'(f)\downarrow$, then $\lim_{n\to\infty} \mathrm{SB}(f[n])\downarrow \geq \mathbf{M}'(f)$; if $\mathbf{M}'(f)\uparrow$, then $\liminf_{n\to\infty} \mathrm{SB}(f[n]) = \infty$. Also $\mathrm{SB}(f[n]) \leq n$.

Now $\mathbf{M}(f[n])$ outputs $\mathrm{Prog}(P_n^f, f[\mathrm{SB}(f[n])])$, where $P_n^f = \{i \leq \mathrm{SB}(f[n]) \mid \varphi_{i,n} \subseteq f\}$.

Claim 33 *If $f \in \mathcal{S}$ then* $\mathbf{M}$ **Ex**-*identifies f.*

PROOF.

Suppose $f \in \mathcal{S}$. Thus, $\lim_{n\to\infty} \mathrm{SB}(f[n])\downarrow$. Let $\mathrm{SB}(f)$ denote $\lim_{n\to\infty} \mathrm{SB}(f[n])$. Note that $\mathrm{SB}(f) \geq \mathrm{MinProg}(f)$. Let n_0 be such that, for all $n \geq n_0$, $\mathrm{SB}(f[n]) = \mathrm{SB}(f)$, and $(\forall p \leq \mathrm{SB}(f) \mid \varphi_p \not\subseteq f)[\varphi_{p,n_0} \not\subseteq f]$. It follows that for all $n \geq n_0$, $P_n^f = \{p \leq \mathrm{SB}(f) \mid \varphi_p \subseteq f\}$. Since $\mathrm{MinProg}(f) \leq \mathrm{SB}(f)$, it follows that $\mathrm{MinProg}(f) \in P_n^f$. Thus, $\mathbf{M}(f)\downarrow = \mathrm{Prog}(P_{n_0}^f, f[\mathrm{SB}(f)])$. Now, using Claim 31 we have that $\varphi_{\mathrm{Prog}(P_n^f, f[\mathrm{SB}(f)])} = f$. Claim follows. $\square$ (Claim 33)

Claim 34 *If $f \in \mathcal{R} - \mathcal{S}$, then* $\mathbf{M}$ $\mathbf{Bc}^*$-*identifies f.*

PROOF. Suppose $f \in \mathcal{R} - \mathcal{S}$. Then, $\liminf_{n\to\infty} \mathrm{SB}(f[n]) = \infty$. Now fix n_0 such that

(i) for all $n \geq n_0$, $\mathrm{SB}(f[n]) \geq \mathrm{MinProg}(f)$, and

(ii) for $n \geq n_0$, for all P, $(\exists p \in P)[\varphi_p = f] \Rightarrow [\varphi_{\mathrm{Prog}(P, f[n])} =^* f]$.

Note that by Claim 32, n_0 satisfying (ii) exists. Note that by (i), and definition of P_n^f, for all $n \geq n_0$, P_n^f contains $\mathrm{MinProg}(f)$.

It follows from (ii) that, for all $n \geq n_0$, $[\varphi_{\mathrm{Prog}(P, f[n])} =^* f]$. This proves the claim. □ (Claim 34)

Theorem follows from Claim 33 and 34. □(Theorem 30)

On the other hand, Corollary 36 below shows that **RelEx** cannot be replaced by finite identification.

Definition 35 [Gol67]

(a) $\mathbf{M}$ *finitely-identifies* f, iff there exists an n and a p such that, $\varphi_p = f$, $\mathbf{M}(f[m]) =?$, for $m < n$, and $\mathbf{M}(f[m]) = p$, for $m \geq n$.

(b) $\mathbf{M}$ finitely-identifies $\mathcal{C}$, iff $\mathbf{M}$ finitely-identifies each $f \in \mathcal{C}$.

(c) $\mathcal{C}$ is finitely-identifiable iff some $\mathbf{M}$ finitely-identifies $\mathcal{C}$.

Let $\mathcal{S}_0 = \{f \mid \varphi_{f(0)} = f\}$. Note that $\mathcal{S}_0$ is finitely-identifiable.

Corollary 36 *Suppose $n \in N$ and $\mathbf{M}$ $\mathbf{Bc}^*$-identifies $\mathcal{R}$. Then there exists an $f \in \mathcal{S}_0$ such that $\mathbf{M}$ does not $\mathbf{Bc}^n$-identify f.*

PROOF. Suppose $\mathbf{M}$ is as given in the hypothesis. Let $g(x) = n$. Then, proof of Theorem 4 gives an $f \in \mathcal{S}_0$, such that for infinitely many x, $\varphi_{\mathbf{M}(f[x])} \neq^n f$. □

It would be interesting to study what other useful properties a suitable general purpose learner can be made to satisfy.

6 Conclusions

Harrington [CS83] surprisingly constructed a general purpose learner, i.e., a machine which $\mathbf{Bc}^*$-identifies all the computable functions. However, the programs output by Harrington's machine become more and more degenerate, i.e., in general, the finite set of anomalies in each final program grows without bound. In this paper we showed that this is unavoidable (Theorem 4 above).

Since the programs output by any general purpose learning machine make large number of errors on infinitely many functions, it is interesting to study how these errors are or can be distributed. Based on this motivation we defined new criteria of inference called $\mathbf{Bc}_m^n$, and completely resolved the relationship between different $\mathbf{Bc}_m^n$ criteria of inference. Among other results, we showed that any general purpose learning machine is poor in predicting near future values. In particular any general purpose learning machine $\mathbf{M}$ predicts the next n values wrongly infinitely often. In contrast, though, we show that the density of such bad prediction points can be made vanishingly small (Theorem 28 above).

We constructed a general purpose learning machine $\mathbf{M}$ such that, on any computable function input, all but finitely many of the programs output by $\mathbf{M}$ are for total functions. Hence, almost all of its conjectures satisfy Popper's Refutability Principle.

We also showed that for every class of computable functions, $\mathcal{S}$, which can be $\mathbf{Ex}$-identified by a reliable machine [Min76,BB75,CJNM94] (see definition in Section 5 above), some general purpose learning machine additionally $\mathbf{Ex}$-identifies $\mathcal{S}$. We further show, though, that reliable identification in the just above statement cannot be replaced by finite identification.

It would be interesting to study which other useful properties a general purpose learner can or cannot have.

References

[Bār71] J. Bārzdiņš. *Complexity and Frequency Solution of Some Algorithmically Unsolvable Problems.* PhD thesis, Novosibirsk State University, 1971. In Russian.

[Bār74] J. Bārzdiņš. Two theorems on the limiting synthesis of functions. In *Theory of Algorithms and Programs, vol. 1*, pages 82–88. Latvian State University, 1974. In Russian.

[BB75] L. Blum and M. Blum. Toward a mathematical theory of inductive inference. *Information and Control*, 28:125–155, 1975.

[Blu67] M. Blum. A machine-independent theory of the complexity of recursive functions. *Journal of the ACM*, 14:322–336, 1967.

[Cas74] J. Case. Periodicity in generations of automata. *Mathematical Systems Theory*, 8:15–32, 1974.

[CJNM94] J. Case, S. Jain, and S. Ngo Manguelle. Refinements of inductive inference by Popperian and reliable machines. *Kybernetika*, 30:23–52, 1994.

[CS83] J. Case and C. Smith. Comparison of identification criteria for machine inductive inference. *Theoretical Computer Science*, 25:193–220, 1983.

[FBW98] R. Freivalds, O. Botuscharov, and R. Wiehagen. Identifying nearly minimal Gödel numbers from additional information. *Annals of Mathematics and Artificial Intelligence*, 23:199–209, 1998.

[FW79] R. Freivalds and R. Wiehagen. Inductive inference with additional information. *Journal of Information Processing and Cybernetics (EIK)*, 15:179–195, 1979.

[Gol67] E. M. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

[JORS99] S. Jain, D. Osherson, J. Royer, and A. Sharma. *Systems that Learn: An Introduction to Learning Theory.* MIT Press, Cambridge, Mass., second edition, 1999.

[Min76] E. Minicozzi. Some natural properties of strong identification in inductive inference. *Theoretical Computer Science*, pages 345–360, 1976.

[Pod74] K. Podnieks. Comparing various concepts of function prediction, Part I. In *Theory of Algorithms and Programs, vol. 1*, pages 68–81. Latvian State University, Riga, Latvia, 1974.

[Pop68] K. Popper. *The Logic of Scientific Discovery.* Harper Torch Books, New York, second edition, 1968.

[Ric80] G. Riccardi. *The Independence of Control Structures in Abstract Programming Systems.* PhD thesis, SUNY/Buffalo, 1980.

[Ric81] G. Riccardi. The independence of control structures in abstract programming systems. *Journal of Computer and System Sciences*, 22:107–143, 1981.

[Rog58] H. Rogers. Gödel numberings of partial recursive functions. *Journal of Symbolic Logic*, 23:331–341, 1958.

[Rog67] H. Rogers. *Theory of Recursive Functions and Effective Computability.* McGraw-Hill, 1967. Reprinted by MIT Press in 1987.

[Roy87] J. Royer. *A Connotational Theory of Program Structure*, volume 273 of *Lecture Notes in Computer Science*. Springer-Verlag, 1987.