ON LEARNING LIMITING PROGRAMS[*]

JOHN CASE[†]
Department of Computer and Information Sciences
University of Delaware
Newark, DE 19716, USA

SANJAY JAIN[‡]
Department of Computer and Information Sciences
University of Delaware
Newark, DE 19716, USA

ARUN SHARMA[§]
School of Computer Science and Engineering
The University of New South Wales
Sydney, NSW 2033
Australia

Abstract

Machine learning of *limit programs* (i.e., programs allowed finitely many mind changes about their legitimate outputs) for *computable* functions is studied. Learning of *iterated* limit programs is also studied. To partially motivate these studies, it is shown that, in some cases, interesting global properties of computable functions can be proved from suitable $(n+1)$-iterated limit programs for them which can *not* be proved from *any* n-iterated limit programs for them. It is shown that learning power is increased when $(n+1)$-iterated limit programs rather than n-iterated limit programs are to be learned. Many trade-off results are obtained regarding learning power, number (possibly zero) of limits taken, program size constraints and information, and number of errors tolerated in final programs learned.

Keywords: Machine Learning; Limiting Programs; Recursion Theory.

[*]Research partially supported by NSF grant CCR 871-3846 at the University of Delaware and a grant from Siemens Corporation in the Department of Brain and Cognitive Sciences, MIT.
[†]email: case@cis.udel.edu
[‡]email: sjain@cis.udel.edu
[§]email: arun@spectrum.cs.unsw.oz.au

1 Preliminaries

1.1 Notation

Any unexplained recursion theoretic notation is from [33]. N denotes the set of natural numbers, $\{0, 1, 2, 3, \ldots\}$. Unless otherwise specified, $c, e, i, j, k, l, m, n, r, s, t, u, v, w, x, y, z$, with or without decorations[a], range over N. $*$ denotes a non-member of N and is assumed to satisfy $(\forall n)[n < * < \infty]$. a and b with or without decorations, range over $N \cup \{*\}$. $\emptyset$ denotes the empty set. $\subseteq$ denotes subset. $\subset$ denotes proper subset. For S, a subset of N, $\mathrm{card}(S)$ denotes the cardinality of S. $\uparrow$ denotes undefined. $\max(\cdot), \min(\cdot)$ denote the maximum and minimum of a set, respectively, where $\max(\emptyset) = 0$ and $\min(\emptyset) =\uparrow$.

η and θ range over *partial* functions with arguments and values from N. $\eta(x)\downarrow$ denotes that $\eta(x)$ is defined; $\eta(x)\uparrow$ denotes that $\eta(x)$ is undefined.

f, g, h, p, q, F and G with or without decorations range over *total* functions with arguments and values from N. For $n \in N$ and partial functions η and θ, $\eta =^n \theta$ means that $\mathrm{card}(\{x \mid \eta(x) \neq \theta(x)\}) \leq n$; $\eta =^* \theta$ means that $\mathrm{card}(\{x \mid \eta(x) \neq \theta(x)\})$ is finite. $\mathrm{domain}(\eta)$ and $\mathrm{range}(\eta)$ denote the domain and range of the function η, respectively.

We say that η is *monotone* $\overset{\mathrm{def}}{\Leftrightarrow}$ $(\forall x, y \mid x < y)[\eta(x)\downarrow < \eta(y)\downarrow]$. Thus η is monotone iff it is a strictly increasing *total* function.

$\langle i, j\rangle$ stands for an arbitrary, computable, one-to-one encoding of all pairs of natural numbers onto N [33] (we assume that $\langle i, j\rangle \geq \max(\{i, j\})$ and that $\langle \cdot, \cdot\rangle$ is increasing in both its arguments). π_1 and π_2 denote the corresponding, computable inverses: for all x, y, $\pi_1(\langle x, y\rangle) = x$ and $\pi_2(\langle x, y\rangle) = y$. Similarly we can define $\langle \cdot, \ldots, \cdot\rangle$ for encoding multiple natural numbers onto N.

φ denotes a fixed *acceptable* programming system for the partial computable functions: $N \to N$ [32, 33, 25]. (Case showed the acceptable systems are *characterized* as those in which every control structure can be constructed; Royer and later Marcoux examined complexity analogs of this characterization [30, 31, 34, 26].) φ_i denotes the partial computable function computed by program i in the φ-system. Φ denotes an arbitrary Blum complexity measure [5, 21] for the φ-system. The set of all total recursive functions of one variable is denoted by $\mathcal{R}$. $\mathcal{C}$, with or without decorations, ranges over subsets of $\mathcal{R}$. $\mathrm{MinProg}^a(f)$ denotes $\min(\{i \mid \varphi_i =^a f\})$. For $a = 0$, we often drop 0 in $\mathrm{MinProg}^0(f)$, i.e., $\mathrm{MinProg}(f) = \mathrm{MinProg}^0(f)$.

[a]Decorations are subscripts, superscripts and the like.

We sometimes consider partial computable functions with multiple arguments in the φ system. In such cases we implicitly assume that a $\langle\cdot,\ldots,\cdot\rangle$ is used to code the arguments, so, for example, $\varphi_i(x,y)$ stands for $\varphi_i(\langle x,y\rangle)$.

A function f is said to be the *zero-extension* of $\eta \overset{\text{def}}{\Leftrightarrow}$

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

EvntlyConst denotes the set of eventually constant functions, i.e., $\{f \mid (\exists c, n)(\forall x > n)[f(x) = c]\}$.

The quantifiers '$\overset{\infty}{\forall}$' and '$\overset{\infty}{\exists}$', essentially from [5], mean 'for all but finitely many' and 'there exist infinitely many', respectively.

1.2 Fundamental Function Inference Paradigms

An *Inductive Inference Machine* (IIM) [20] is an algorithmic device which takes as its input a set of data given one element at a time, and which from time to time, as it is receiving its input, outputs programs. IIMs have been used in the study of machine identification of programs for computable functions as well as algorithmic learning of grammars for languages [4, 10, 11, 18, 20, 28, 40].[b] For a survey of this work see [1, 28, 23, 7].

M, with or without decorations, ranges over the class of inductive inference machines. For inference of a computable function f by an IIM **M**, the graph of f is fed to **M** in any order. Without loss of generality [4, 10], we will assume that **M** is fed the graph of f in the sequence $(0, f(0)), (1, f(1)), (2, f(2)), \ldots$. For all computable functions f, $f[n]$ denotes the finite initial segment $((0, f(0)), (1, f(1)), \ldots, (n-1, f(n-1)))$. Let INIT $= \{f[n] \mid f \in \mathcal{R} \wedge n \in N\}$. Variables σ and τ, with or without decorations, range over INIT. $|\tau|$ denotes the number of elements in τ. Thus $|f[n]| = n$. content(τ) denotes the set of pairs in the range of τ. Thus content$(f[n]) = \{(i, f(i)) \mid i < n\}$. $\mathbf{M}(\sigma)$ is the last output of **M** by the time it has received input σ. We will assume, without loss of generality, that $\mathbf{M}(\sigma)$ is always defined. We say that $\mathbf{M}(f)$ converges to i (written: $\mathbf{M}(f)\!\downarrow = i$) iff $(\overset{\infty}{\forall} n)[\mathbf{M}(f[n]) = i]$; $\mathbf{M}(f)$ is undefined if no such i exists.

[b]The separation, but not the strong separation results in the present paper go through for language learning. We have not yet investigated language learning analogs of most of our other results.

Definition 1 [20, 4, 10] Suppose $a \in N \cup \{*\}$.

(a) $\mathbf{M}$ $\mathbf{Ex}^a$-*identifies* a computable function $f \in \mathcal{R}$ (written: $f \in \mathbf{Ex}^a(\mathbf{M})$) iff both $\mathbf{M}(f)\downarrow$ and $\varphi_{\mathbf{M}(f)} =^a f$.

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

Case and Smith [10] introduced another infinite hierarchy of identification criteria which we describe below. "**Bc**" stands for *behaviorally correct.* Barzdin [3] essentially introduced the notion of $\mathbf{Bc}^0$.

Definition 2 [10] Suppose $a \in N \cup \{*\}$.

(a) $\mathbf{M}$ $\mathbf{Bc}^a$-*identifies* a computable function $f \in \mathcal{R}$ (written: $f \in \mathbf{Bc}^a(\mathbf{M})$) iff $(\overset{\infty}{\forall} n)[\varphi_{\mathbf{M}(f[n])} =^a f]$.

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

We usually write **Ex** for $\mathbf{Ex}^0$ and **Bc** for $\mathbf{Bc}^0$. Theorem 3 just below states some of the basic hierarchy results about the $\mathbf{Ex}^a$ and $\mathbf{Bc}^a$ classes.

Theorem 3 *For all $n \in N$,*

(a) $\mathbf{Ex}^n \subset \mathbf{Ex}^{n+1}$*,*

(b) $\bigcup_{n\in N} \mathbf{Ex}^n \subset \mathbf{Ex}^*$*,*

(c) $\mathbf{Ex}^* \subset \mathbf{Bc}$*,*

(d) $\mathbf{Bc}^n \subset \mathbf{Bc}^{n+1}$*,*

(e) $\bigcup_{n\in N} \mathbf{Bc}^n \subset \mathbf{Bc}^*$*, and*

(f) $\mathcal{R} \in \mathbf{Bc}^*$*.*

Parts (a), (b), (d), and (e) are due to Case and Smith [10]. John Steel first observed that $\mathbf{Ex}^* \subseteq \mathbf{Bc}$ and diagonalization in part (c) is due to Harrington and Case [10]. Part (f) is due to Harrington [10]. Blum and Blum [4] first showed that $\mathbf{Ex} \subset \mathbf{Ex}^*$. Barzdin [3] independently showed $\mathbf{Ex} \subset \mathbf{Bc}$.

2 Limiting Programs

2.1 Definition and Motivation of Limiting Programs

For each i, consider the following corresponding procedure for "computing" a (partial) function φ_i^1.

```
On input $x$
    for $t = 0$ to $\infty$
    Start a new clone of $\varphi$-program $i$ running on input $(x, t)$
    endfor
```

It is to be understood that

1. each iterate of the **for**-loop finishes since it merely *starts* a process running and
2. in some iterates of the **for**-loop the process *started* may itself never converge.

$\varphi_i^1(x) \overset{\text{def}}{=}$ the unique y (if any) eventually output by all but finitely many of the clones of φ-program i in the **for**-loop above. Equivalently, $\varphi_i^1(x) \overset{\text{def}}{=} \lim_{t\to\infty} \varphi_i(x, t)$.

We shall refer to i as $\mathbf{Lim}^1$-program i (in the φ^1-system) when we are thinking of i as encoding the **for**-loop above rather than as encoding φ-program i.

Intuitively, $\mathbf{Lim}^1$-program i (in the φ^1-system) is a procedure, which on an input for which it has an output, is allowed to change its mind finitely many times about that output (or even about whether to output at all). N.B. there may be no algorithm for signaling when a $\mathbf{Lim}^1$-program has stopped changing its mind about its output.

The partial functions which are the limit of some *total* computable function are well known to be characterized as exactly the partial functions computable relative to an oracle for the halting problem [37, 29, 19, 38, 39].[c] This result and its relativizations were first noticed and used by Post [36] and have been employed (sometimes with rediscovery) many times. [24] studied acceptable programming systems for partial functions computable relative to oracles. Most of the results of this paper would hold also for such programming systems, but we will present our results directly about systems such as φ^1.

The creation of clones as in $\mathbf{Lim}^1$-program i above can be iterated. This motivates the following.

$$\varphi_i^n(x) \overset{\text{def}}{=} \lim_{t_1\to\infty} \lim_{t_2\to\infty} \ldots \lim_{t_n\to\infty} \varphi_i(x, t_1, t_2, \ldots, t_n)$$

We will sometimes say that i is a $\mathbf{Lim}^n$*-program* for φ_i^n.

[c]The class of partial functions which are the limit of some *partial* recursive function, i.e., $\{\varphi_i^1 \mid i \in N\}$, is a larger class than the class of partial functions computable in the halting problem.

In this paper we shall be interested in $\mathbf{Lim}^n$-programs (from the φ_i^n system) which happen to compute *partial computable* functions! We study the learning of $\mathbf{Lim}^n$-programs for *computable* functions. The $n = 0$ case is merely the special case studied previously in the literature. We are interested in, among other things, the tradeoffs between the size of n and the resultant ability to learn large classes of computable functions. We shall see that, many times, larger classes can be learned for $n + 1$ limits than for n.

The reader may be thinking that, for $n > 0$, $\mathbf{Lim}^n$-programs are not particularly useful. What does it profit one to have learned, say, a $\mathbf{Lim}^1$-program for an $f \in \mathcal{R}$? Well, with such a program one can discover values for f eventually (albeit without knowing when one has found those values), but it is easy to argue that "eventually" is too long to wait.

Actually, $\mathbf{Lim}^n$-programs, for $n > 0$, can be quite useful.

In physics it is sometimes easier to infer a *global* property of a phenomenon than it is to make more detailed predictions about observations; for example, Kepler's Laws are easier to derive than equations of motion of planets. In the next section (Section 2.2) we show that it is, in some cases, possible to prove *global* properties of a computable function from a suitable $\mathbf{Lim}^1$-program for it when it is *not* possible to prove these properties from *any* of the ordinary (φ or $\mathbf{Lim}^0$) programs for it.

2.2 Further Motivation

We next provide the preliminaries for obtaining the provability results as advertised just above at the end of previous section (Section 2.1).

We shall present our results for extensions of first order arithmetic. Regarding expressing propositions in first order arithmetic, we shall proceed informally. If E is an expression such as '$\Phi_i \leq t$', or 'φ_i is total', we shall write $\ll \mathsf{E} \gg$ to denote a naturally corresponding, fixed standard wff of first order arithmetic [27] which (semantically) expresses E. We need and assume that

> if E' is obtained from E by changing some numerical values, then $\ll \mathsf{E}' \gg$ can be *algorithmically* obtained from those changed numerical values and $\ll \mathsf{E} \gg$.

It is understood that, if E contains references to partial functions, such as φ and Φ, then in $\ll \mathsf{E} \gg$ these are, in effect, named by standard programs for them. It is well known that wffs extensionally equivalent (with respect to standard models) may not be intensionally or provably equivalent [14]. In what follows, when we use the $\ll \mathsf{E} \gg$ notation, it will always be for propositions that are easily seen to be (semantically) expressible in first order arithmetic.[d] '$\vdash$' denotes the provability relation.

Theorem 4 *Suppose* $\mathbf{T}$ *is an axiomatizable (i.e., r.e.* [13]*) first order theory which extends Peano Arithmetic* [27] *and in which one can*not *prove anything false about totality of (partial) computable functions computed by programs in* φ*. Then there are* e_0 *and* e_1 *such that* $\varphi_{e_0} = \varphi^1_{e_1}$ *is total, yet*

1. $(\forall i \mid \varphi_i = \varphi_{e_0})[\mathbf{T} \not\vdash \ll \varphi_i \textit{ is total } \gg]$, and
2. $[\mathbf{T} \vdash \ll \varphi^1_{e_1} \textit{ is total} \gg]$.

Proof. Suppose the hypotheses. Fix an automatic theorem prover for $\mathbf{T}$. In what follows any reference to proving something in $\mathbf{T}$ within so many steps refers to steps in the execution of this automatic theorem prover.

Clearly there is a φ-program e_0 such that, for all x,

$$\varphi_{e_0}(x) = 1 + \max(\{\varphi_i(x) \mid i \leq x \ \wedge\ [\mathbf{T} \vdash \ll \varphi_i \text{ is total} \gg] \text{ within } x \text{ steps}\}).$$

Clearly φ_{e_0} is total and e_0 satisfies the first clause in the statment of Theorem 4.

Also, there is a program e_1 such that, for all x,

$$\varphi_{e_1}(x,t) = \begin{cases} 0 & \text{if } \Phi_{e_0}(x) > t; \\ \varphi_{e_0}(x) & \text{otherwise.} \end{cases}$$

A simple case analysis proves that $\varphi^1_{e_1} = \lim_{t\to\infty} \varphi_{e_1}(\cdot, t)$ is total. Furthermore, *this* proof of totality is clearly formalizable in Peano Arithmetic, hence, in $\mathbf{T}$.

Since φ_{e_0} is total, $\varphi^1_{e_1} = \varphi_{e_0}$. ▮

The variant of Theorem 4 above in which 'totality' and 'total' are replaced by 'monotonicity' and 'monotone', respectively can clearly be proved by a slightly more complex construction than that in the proof just above of Theorem 4. Monotonicity is clearly an interesting global property of computable functions.

Also we can easily prove a variant of each of these theorems in which φ-programs and φ^1-programs are replaced by φ^n-programs and φ^{n+1}-programs, respectively.

In a sense we may consider the $\mathbf{Lim}^n$-programs to be *higher order*. [2] contains many results about learning a different class of higher order programs with comparisons to some of those in the present paper.

[d]These informal discussion of provability and expressibility are based on Section 4.3 of [35].

3 Learning Limiting Programs

3.1 Definitions

Definition 5 Let $a \in N \cup \{*\}, n \in N$.

(a) $\mathbf{M}$ $\mathbf{Lim}^n\mathbf{Ex}^a$-identifies $f \in \mathcal{R}$ (written: $f \in \mathbf{Lim}^n\mathbf{Ex}^a(\mathbf{M})$) iff $\mathbf{M}(f)\downarrow$ and $\varphi^n_{\mathbf{M}(f)} =^a f$.

(b) $\mathbf{Lim}^n\mathbf{Ex}^a = \{\mathcal{C} \subseteq \mathcal{R} \mid (\exists \mathbf{M})[\mathcal{C} \subseteq \mathbf{Lim}^n\mathbf{Ex}^a(\mathbf{M})]\}$.

Definition 6 Let $a \in N \cup \{*\}, n \in N$.

(a) $\mathbf{M}$ $\mathbf{Lim}^*\mathbf{Ex}^a$-identifies $f \in \mathcal{R}$ (written: $f \in \mathbf{Lim}^*\mathbf{Ex}^a(\mathbf{M})$) iff $\mathbf{M}(f)\downarrow = i$ such that $\varphi^{\pi_1(i)}_{\pi_2(i)} =^a f$.

(b) $\mathbf{Lim}^*\mathbf{Ex}^a = \{\mathcal{C} \subseteq \mathcal{R} \mid (\exists \mathbf{M})[\mathcal{C} \subseteq \mathbf{Lim}^*\mathbf{Ex}^a(\mathbf{M})]\}$.

We often denote $\mathbf{Lim}^1\mathbf{Ex}^a$ by $\mathbf{LimEx}^a$ ($\mathbf{Lim}^1\mathbf{Ex}^0$ by $\mathbf{LimEx}$) and $\mathbf{Lim}^a\mathbf{Ex}^0$ by $\mathbf{Lim}^a\mathbf{Ex}$.

Definition 7 Let $a \in N \cup \{*\}$.

(a) $\mathbf{M}$ $\mathbf{LimBc}^a$-identifies $f \in \mathcal{R}$ (written: $f \in \mathbf{LimBc}^a(\mathbf{M})$) iff $(\overset{\infty}{\forall} n)[\varphi^1_{\mathbf{M}(f[n])} =^a f]$.

(b) $\mathbf{LimBc}^a = \{\mathcal{C} \subseteq \mathcal{R} \mid (\exists \mathbf{M})[\mathcal{C} \subseteq \mathbf{LimBc}^a(\mathbf{M})]\}$.

We usually write **LimBc** for $\mathbf{LimBc}^0$. We do not consider iterated limits for **Bc**-identification since $\mathcal{R} \in \mathbf{LimBc}$ (Theorem 14) and thus iterating limits does not help.

3.2 Results

The next result implies that allowing an extra anomaly can sometimes more than makeup for any finite number of limits.

Theorem 8 $(\forall i)[\mathbf{Ex}^{i+1} - \mathbf{Lim}^*\mathbf{Ex}^i \neq \emptyset]$.

Proof. Fix i. Consider the following class of functions:

$\mathcal{C} = \{f \mid \varphi_{f(0)} =^{i+1} f\}$.

Clearly, $\mathcal{C}$ is in $\mathbf{Ex}^{i+1}$. Suppose by way of contradiction machine $\mathbf{M}$ $\mathbf{Lim}^*\mathbf{Ex}^i$ identifies $\mathcal{C}$. Then by the Kleene recursion theorem [33, Page 214] there exists an e such that the (partial) function φ_e may be defined as follows.

Let $\varphi_e(0) = e$. Let x_s denote the least x such that $\varphi_e(x)$ has not been defined before stage s. Thus $x_0 = 1$. Go to stage 0.

```
Begin stage $s$

1.  For each $y \in N$, let $f_y$ denote the function defined as follows.
        $$f_y(x) = \begin{cases} \varphi_e(x) & x < x_s; \\ y & x_s \leq x \leq x_s + i; \\ 0 & \text{otherwise.} \end{cases}$$

2.  Dovetail steps 3 and 4 in parallel until step 4 succeeds. If and when step 4
      succeeds, go to step 5.

3.  Let $x = x_s + i + 1$.
    repeat
        $\varphi_e(x) = 0$; $x = x + 1$.
    forever

4.  Search for a $y \in N$ and $n > x_s + i + 1$ such that, $\mathbf{M}(\varphi_e[x_s]) \neq \mathbf{M}(f_y[n])$.

5.  If and when 4 above succeeds, let $y$ and $n$ be as found in step 4. For $x_s \leq x \leq$
      $x_s + i$, let $\varphi_e(x) = y$. For $x_s + i < x < n$ such that $\varphi_e$ has not been defined
      till now, let $\varphi_e(x) = 0$.

6.  Go to stage $s + 1$.

End stage $s$
```

Now consider the following cases:

Case 1: Infinitely many stages are executed.

In this case, let $f = \varphi_e \in \mathcal{C}$. Clearly, $\mathbf{M}(f)\uparrow$ (because of the success of step 4 in each stage s).

Case 2: Some stage s starts but never finishes.

Let stage s be the stage which starts but never finishes. For each y, let f_y be as defined in step 1 of stage s.

Now for all y, $f_y \in \mathcal{C}$. Also for each y, $\mathbf{M}(f_y)\downarrow = \mathbf{M}(\varphi_e[x_s])$. But $\pi_2(\mathbf{M}(\varphi_e[x_s]))$ can be a $\mathbf{Lim}^{\pi_1(\mathbf{M}(\varphi_e[x_s]))}$ program for an i variant of at most finitely many f_y. Thus there exists an f_y ($\in \mathcal{C}$) such that $\mathbf{M}$ does not $\mathbf{Lim}^*\mathbf{Ex}^i$ identify f_y.

From the above cases it follows that $\mathcal{C} \notin \mathbf{Lim}^*\mathbf{Ex}^i$. ∎

Similarly it can be shown that $\mathcal{C} = \{f \mid \varphi_{f(0)} =^* f\} \in \mathbf{Ex}^* - \bigcup_i \mathbf{Lim}^*\mathbf{Ex}^i$.

Theorem 9 $\mathbf{Ex}^* - \bigcup_{i \in N} \mathbf{Lim}^*\mathbf{Ex}^i \neq \emptyset$.

As a corollary to Theorems 8 and 9 we have

Corollary 10 $(\forall a \in N \cup \{*\})[\mathbf{Lim}^a\mathbf{Ex}^0 \subset \mathbf{Lim}^a\mathbf{Ex}^1 \subset \mathbf{Lim}^a\mathbf{Ex}^2 \subset \cdots \subset \mathbf{Lim}^a\mathbf{Ex}^i \subset \mathbf{Lim}^a\mathbf{Ex}^{i+1} \subset \cdots \subset \mathbf{Lim}^a\mathbf{Ex}^*]$.

A modification (which is similar in nature to the proof of Theorem 8) to the proof of $\mathbf{Bc} - \mathbf{Ex}^* \neq \emptyset$ in [10] can be used to show that:

Theorem 11 $\mathbf{Bc} - \mathbf{Lim}^*\mathbf{Ex}^* \neq \emptyset$.

Theorem 12 $(\forall j)[\mathbf{LimEx} - \mathbf{Bc}^j \neq \emptyset]$.

Proof. Consider the following class of functions:

$\mathcal{C} = \{f \in \mathcal{R} \mid f$ is the zero-extension of $\varphi_{f(0)}\}$.

It is easy to see that $\mathcal{C} \in \mathbf{LimEx}$. Suppose by way of contradiction that $\mathbf{M}$ $\mathbf{Bc}^j$ identifies $\mathcal{C}$. Then by the Kleene recursion theorem [33] there exists an e such that the (partial) function φ_e may be defined as follows.

Let $\varphi_e(0) = e$. Let x_s denote the least x such that $\varphi_e(x)$ has not been defined before stage s. Go to stage 0.

Begin stage s

1. Let f be the zero-extension of $\varphi_e[x_s]$. Search for $y_1, y_2, \ldots, y_{j+1}, n$ such that $y_{j+1} > y_j > \ldots > y_1 > n > x_s$ and $\varphi_{\mathbf{M}(f[n])}(y_i)\downarrow$, for each i such that $1 \leq i \leq j+1$.
2. If and when such $y_i's$ and n are found let:

   $\varphi_e(y_i) = \varphi_{\mathbf{M}(f[n])}(y_i) + 1$, for $1 \leq i \leq j+1$.

   $\varphi_e(x) = 0$, for $x_s \leq x < y_{j+1}$ and $x \notin \{y_1, y_2, \ldots, y_{j+1}\}$.
3. Go to stage $s+1$.

End stage s

Now consider the following cases:

Case 1: Infinitely many stages are executed.

In this case let $f = \varphi_e \in \mathcal{C}$. However, $\varphi_{\mathbf{M}(f[n])} \neq^j f$ for infinitely many n (by the success of the search in step 1, in each stage s, and the diagonalization in step 2).

Case 2: Some stage s starts but never finishes.

Let stage s be the stage which starts but never finishes. Let f be the zero-extension of φ_e. Now for all but finitely many n, for all but finitely many x, $\varphi_{\mathbf{M}(f[n])}(x)\uparrow$ (otherwise step 1 would succeed).

From the above cases it follows that $\mathcal{C} \notin \mathbf{Bc}^j$. ▮

As a corollary we have:

Corollary 13 $(\forall a \in N \cup \{*\})[\mathbf{Ex}^a \subset \mathbf{LimEx}^a]$.

Theorem 14 $\mathcal{R} \in \mathbf{LimBc}$.

Proof. Let $\mathbf{M}(f[n])$=i such that,

$$\varphi_i(x,j) = \begin{cases} \varphi_k(x), & \text{if } (\exists l \leq j)(\forall y < n)[\Phi_l(y) \leq j \wedge \varphi_l(y) = f(y) \wedge \Phi_l(x) \leq j] \bigwedge \\ & \quad k \text{ is the least such } l; \\ 0, & \text{otherwise.} \end{cases}$$

It is easy to see that $\mathbf{M}$ witnesses that $\mathcal{R} \in \mathbf{LimBc}$. ▮

The next Theorem implies that allowing an extra limit can sometimes more than makeup for allowing an arbitrary but finite number of anomalies.

Theorem 15 $(\forall n)[\mathbf{Lim}^{n+1}\mathbf{Ex} - \mathbf{Lim}^n\mathbf{Ex}^* \neq \emptyset]$.

Proof. Consider the following class of functions:

$\mathcal{C}_n = \{f \in \mathcal{R} \mid f(0)$ is a $\mathbf{Lim}^{n+1}$-program for $f\}$.

Clearly, $\mathcal{C}_n \in \mathbf{Lim}^{n+1}\mathbf{Ex}$. We show that $\mathcal{C}_1 \notin \mathbf{Lim}^1\mathbf{Ex}^*$. Proof can be easily extended to show that $\mathcal{C}_n \notin \mathbf{Lim}^n\mathbf{Ex}^*$.

Suppose by way of contradiction that $\mathcal{C}_1 \subseteq \mathbf{Lim}^1\mathbf{Ex}(\mathbf{M})$. Then by the Kleene recursion theorem [33] there exists an e such that the (partial) function φ_e may be defined as follows.

We will construct φ_e in stages. e will be a $\mathbf{Lim}^2$-program for a function in $\mathcal{C}_1 - \mathbf{Lim}^1\mathbf{Ex}^*(\mathbf{M})$. Let $f = \varphi_e^2$ (we will argue below that φ_e^2 is indeed a total recursive function).

Let $(\forall t_1, t_2)[\varphi_e(0, t_1, t_2) = e]$. Let x_s denote the least x such that $f(x)$ does not become known before stage s. Go to stage 0.

Begin stage s

1. Dovetail steps 2 and 3 until step 2 succeeds. If and when step 2 succeeds, go to step 4.
2. Search for an extension τ of $f[x_s]$ such that $\mathbf{M}(f[x_s]) \neq \mathbf{M}(\tau)$.
3. Dovetail steps 3.x $(x \geq x_s)$ in such a way that if step 2 does not succeed, then each step gets infinite time.

```
3.x (Note: Basically in this step we try to define $\varphi_e^2(x)$. If step 2 does not succeed
      then this step will be successful in defining $\varphi_e^2(x)$. Assuming that step 2
      does not succeed: $\varphi_e^2(x)$ will be 0 if there exists a $y \geq x$ such that $(\overset{\infty}{\forall}$
      $t)[\varphi_{\mathbf{M}(f[x_s])}(y,t) \neq 0]$; otherwise $\varphi_e^2(x)$ will be 1).
    Go to substage $x$:0.
    Begin Substage $x : \langle y, t\rangle$
        If $y < x$ then go to substage $x : (\langle y, t\rangle + 1)$.
        Let $t_1$ be the least $t_1'$ such that there exists a $t_2'$, for which $\varphi_e(x, t_1', t_2')$ has
          not been defined till now.
        Dovetail step 3.$x$.1 and 3.$x$.2 until 3.$x$.1 succeeds. If and when step 3.$x$.1
          succeeds, go to step 3.$x$.3.
    3.$x$.1 Search for a $t' > t$ such that $\varphi_{\mathbf{M}(f[x_s])}(y, t') = 0$.
    3.$x$.2 Let $w = 0$.
        **repeat**
              Let $u, v$ be such that $w = \langle u, v\rangle$.
              If $\varphi_e(x, u, v)$ has not been defined till now, then let $\varphi_e(x, u, v) = 0$.
              Let $w = w + 1$.
        **forever**
    3.$x$.3 For all $t_2$ such that $\varphi_e(x, t_1, t_2)$ is not defined till now, let $\varphi_e(x, t_1, t_2) = 1$.
        Go to substage $x : (\langle y, t\rangle + 1)$.
    End Substage $x : \langle y, t\rangle$
4.  For $x$ such that $x_s \leq x < |\tau|$, let
        $\varphi_e(x, t_1, t_2) = y$, where $(x, y) \in \mathrm{content}(\tau)$ and $\varphi_e(x, t_1, t_2)$ is not defined
          till now. (Thus $f(x) = y$).
    Go to stage $s + 1$.
    (Note that $\mathbf{M}(f[x_s] \neq f[x_{s+1}])$).
End stage $s$
```

Now consider the following cases:

Case 1: Infinitely many stages are executed.

In this case clearly, f is recursive. Also $\mathbf{M}(f)\uparrow$ (by the success of step 2, in each stage s).

Case 2: Some stage s starts but never finishes.

Let stage s be the stage which starts but never finishes. Now, on all extensions of $f[x_s]$, $\mathbf{M}$ converges to $\mathbf{M}(f[x_s])$. We claim that for $x \geq x_s$, if $(\exists y \geq x)[(\overset{\infty}{\forall} t)[\varphi_{\mathbf{M}(f[x_s])}(y,t) \neq 0]]$, then $f(x) = 0$; $f(x) = 1$ otherwise. This is so because if there exist y, t such that $y \geq x$ and $[(\forall t' > t)[\varphi_{\mathbf{M}(f[x_s])}(y, t') \neq 0]]$, then substage $x : \langle y, t \rangle$, would never end, and thus, due to step 3.x.2, $f(x) = 0$;

if for all $y \geq x$, $[(\overset{\infty}{\exists}\ t)[\varphi_{\mathbf{M}(f[x_s])}(y, t) = 0]]$, then for all n, substage $x : n$ halts and thus $f(x) = 1$.

Thus clearly, $f(x)$ is 1 for all but finitely many x or $f(x)$ is 0 for all but finitely many x. Thus f is recursive. Also $f \neq^* \varphi^1_{\mathbf{M}(f[x_s])}$.

From the above cases it follows that $\mathbf{M}$ does not $\mathbf{Lim}^1\mathbf{Ex}$-identify f. ▮

Similarly we can also prove

Theorem 16 $\mathbf{Lim}^*\mathbf{Ex} - \bigcup_n \mathbf{Lim}^n\mathbf{Ex}^* \neq \emptyset$.

As a corollary to Theorems 15 and 16 we have

Corollary 17 $(\forall a \in N \cup \{*\})[\mathbf{Ex}^a \subset \mathbf{Lim}^1\mathbf{Ex}^a \subset \mathbf{Lim}^2\mathbf{Ex}^a \subset \cdots \subset \mathbf{Lim}^i\mathbf{Ex}^a \subset \mathbf{Lim}^{i+1}\mathbf{Ex}^a \subset \cdots \subset \mathbf{Lim}^*\mathbf{Ex}^a]$.

The following corollary follows from Theorems 8, 9, 15 and 16.

Corollary 18 $(\forall a, b, a', b' \in N \cup \{*\})[[\mathbf{Lim}^a\mathbf{Ex}^b \subseteq \mathbf{Lim}^{a'}\mathbf{Ex}^{b'}] \Leftrightarrow [a \leq a' \wedge b \leq b']]$.

4 Nearly Minimal Identification of Limiting Programs

Freivalds and Chen [15, 12, 11] studied the learning of nearly minimal size programs. Here we study limiting analogs.

Definition 19 Let $a, b \in N \cup \{*\}$ and $h \in \mathcal{R}$.

a) $\mathbf{M}$ h-$\mathbf{MLim}^a\mathbf{Ex}^b$-identifies $f \in \mathcal{R}$ (written: $f \in h$-$\mathbf{MLim}^a\mathbf{Ex}^b(\mathbf{M})$) iff $\mathbf{M}$ $\mathbf{Lim}^a\mathbf{Ex}^b$-identifies f and $\mathbf{M}(f) \leq h(\text{MinProg}(f))$.

b) h-$\mathbf{MLim}^a\mathbf{Ex}^b = \{\mathcal{C} \subseteq \mathcal{R} \mid (\exists \mathbf{M})[\mathcal{C} \subseteq h\text{-}\mathbf{MLim}^a\mathbf{Ex}^b(\mathbf{M})]\}$.

c) $\mathbf{MLim}^a\mathbf{Ex}^b = \bigcup_{h \in \mathcal{R}} h\text{-}\mathbf{MLim}^a\mathbf{Ex}^b$.

We now show the following surprising theorem.

Theorem 20 $(\forall a, b \in N \cup \{*\} \mid a \neq 0)[\mathbf{MLim}^a\mathbf{Ex}^b = \mathbf{Lim}^a\mathbf{Ex}^b]$.

PROOF. We prove the theorem for $a = 1$. Other cases are similar. Suppose $\mathbf{M}$ is given.

Let $m_{i,t} = \min(\{t\} \cup \{x \mid \Phi_i(x) > t\})$. Let g be such that, $\varphi_{g(i)}(x,t) = \varphi_{\mathbf{M}(\varphi_i[m_{i,t}])}(x,t)$.

Let $h(x) = \sum_{k=0}^{x} g(k)$.

Now consider the following machine $\mathbf{M}'$.

$\mathbf{M}'(f[n]) = g(i_{f[n]})$, where $i_{f[n]} = \min(\{n\} \cup \{i \mid i \leq n \wedge \varphi_i[m_{i,n}] = f[m_{i,n}] \wedge \mathbf{M}(f[m_{i,n}]) = \mathbf{M}(f[n])\})$.

Now suppose $f \in \mathbf{Lim}^1\mathbf{Ex}^b(\mathbf{M})$. Clearly for large enough n, $i_{f[n]} = \min(\{i \mid \varphi_i = f \vee [\lim_{t\to\infty} m_{i,t}$ exists and $\varphi_i[\lim_{t\to\infty} m_{i,t}] = f[\lim_{t\to\infty} m_{i,t}] \wedge \mathbf{M}(f) = \mathbf{M}(f[\lim_{t\to\infty} m_{i,t}])]\})$. It follows that for large enough n, $\varphi^1_{\mathbf{M}'(f)} = \varphi^1_{g(i_{f[n]})} = \varphi^1_{\mathbf{M}(f)}$. Thus $f \in \mathbf{Lim}^1\mathbf{Ex}^b(\mathbf{M}')$. Let $i_f = \min(\{i \mid \varphi_i = f \vee [\lim_{t\to\infty} m_{i,t}$ exists and $\varphi_i[\lim_{t\to\infty} m_{i,t}] = f[\lim_{t\to\infty} m_{i,t}] \wedge \mathbf{M}(f) = \mathbf{M}(f[\lim_{t\to\infty} m_{i,t}])]\})$. Now $\mathbf{M}'(f) = g(i_f) \leq h(\mathrm{MinProg}(f))$. Thus $\mathbf{Lim}^1\mathbf{Ex}^b(\mathbf{M}) \subseteq h$-$\mathbf{MLim}^1\mathbf{Ex}^b(\mathbf{M}')$. ▮

5 Using Limiting Programs for Succinctness

Next we show that using **Lim**-programs to infer functions may considerably succinctify ([9]) the final result over using ordinary (φ) programs.

Theorem 21 $(\exists \mathcal{C} \subseteq \mathbf{EvntlyConst} \mid \mathrm{card}(\mathcal{C}) = \infty)(\exists \mathbf{M})(\forall h \in \mathcal{R})[\mathcal{C} \subseteq \mathbf{LimEx}(\mathbf{M}) \wedge (\overset{\infty}{\forall} f \in \mathcal{C})[h(\mathbf{M}(f)) \leq \mathrm{MinProg}^*(f)]]$.

Moreover, in the immediately above Theorem (Theorem 21) we can take $\mathcal{C}$ to be a set of functions computed by an r.e. sequence of $\mathbf{Lim}^1$-programs.

Proof. Using the operator recursion theorem [6] we will describe a sequence of programs $p(0), p(1), \ldots$. Let $f_i = \varphi^1_{p(i)}$ (it will be clear that f_i is a total recursive function; in fact $f_i \in \mathbf{EvntlyConst}$). We let $\mathcal{C} = \{f_i \mid i \in N\}$. We now describe the construction of $\varphi_{p(i)}$. For all t, let $\varphi_{p(i)}(0,t) = i$ (thus $f_i(0) = i$). Go to stage 1.

Definition of $\varphi_{p(i)}$

```
Begin stage $s$
1.  For each $t < s$, let $\varphi_{p(i)}(s,t) = 0$.
2.  Let $m_s = 1 + \max(\{\varphi_j(p(i)) \mid j \leq i \wedge \Phi_j(p(i)) \leq s\})$.
3.  For $k, l \leq m_s$, let
        $w^s_{l,k} = \mathrm{card}(\{x < s \mid [\mathrm{card}(\{j < m_s \mid \Phi_j(x) \leq s\}) = m_s - k] \wedge [l =$
          $\min(N - \{\varphi_j(x) \mid j < m_s \wedge x < s \wedge \Phi_j(x) \leq s\})]\})$.
    For $k, l \leq m_s$, let $c^s_{l,k} = \sum_{j \leq m_s, r < k} w^s_{j,r} + \sum_{j<l} w^s_{j,k}$.
```

```
4.    for $k = 0$ to $m_s$
          for $r = 0$ to $m_s$
              for $l = 1$ to $w^s_{r,k}$
                  $\varphi_{p(i)}(c^s_{r,k} + l, s) = r$
              endfor
          endfor
      endfor
5.    Go to stage $s + 1$.
End stage $s$
End Definition of $\varphi_{p(i)}$
```

Clearly all stages halt. Note that for all but finitely many s, $m_s = 1 + \max(\{\varphi_j(p(i)) \mid j \leq i \wedge \varphi_j(p(i))\downarrow\})$. Let $m = 1+\max(\{\varphi_j(p(i)) \mid j \leq i \wedge \varphi_j(p(i))\downarrow\})$. For (the finitely many) s such that $m_s < m$, $w^s_{l,k}$ may not defined for some values of $l, k \leq m$. To simplify our exposition we assume that these $w^s_{l,k}$ are taken to be 0 in the analysis below. Since $\sum_{k=1}^{m_s} \sum_{l=1}^{m_s} w^s_{l,k} = s$, we have that there exist $k, l \leq m$ such that the sequence $w^0_{l,k}, w^1_{l,k}, w^2_{l,k}, \ldots$ diverges. Let k ($\leq m$) be the least value such that there exists an $l \leq m$ such that the sequence $w^0_{l,k}, w^1_{l,k}, w^2_{l,k}, \ldots$ diverges. Let l ($\leq m$) be the least value such that $w^0_{l,k}, w^1_{l,k}, w^2_{l,k}, \ldots$ does not converge. Now let s be the least value such that for all $s' > s$,

$m_{s'} = m_s$;

$w^{s'}_{l',k'} = w^s_{l',k'}$, for $k' < k$, $l' \leq m$ and

$w^{s'}_{l',k} = w^s_{l',k}$ for $l' < l$.

Now it is easy to see that, for all $s' > s$, $w^{s'+1}_{l,k} \geq w^{s'}_{l,k}$. Thus we have that, for all $x > c^s_{l,k}$, $f_i(x) = l$. Also, it is easy to see that for all $x \leq c^s_{l,k}$, $f_i(x)\downarrow$. Moreover, for all $j \leq m$, there exist infinitely many x such that $\varphi_j(x) \neq l$.

Now let $\mathbf{M}$ be a machine such that for all f, $\mathbf{M}(f[0]) = 0$ and $\mathbf{M}(f[n]) = p(f(0))$, for $n > 0$. Clearly $\mathcal{C} \in \mathbf{LimEx}(\mathbf{M})$. Let $h \in \mathcal{R}$ be given. Let j be such that $\varphi_j = h$. Then for all $i > j$, for all $r \leq h(p(i))$, φ_r is infinitely different from f_i. ▮

As a corollary to the above theorem we have

Corollary 22 $(\exists \mathcal{C} \subseteq \mathbf{EvntlyConst} \mid \mathrm{card}(\mathcal{C}) = \infty)(\exists \mathbf{M})(\forall h \in \mathcal{R})(\forall \mathbf{M}' \mid \mathcal{C} \subseteq \mathbf{Ex}^*(\mathbf{M}'))[\mathcal{C} \subseteq \mathbf{LimEx}(\mathbf{M}) \wedge (\overset{\infty}{\forall} f \in \mathcal{C})[h(\mathbf{M}(f)) \leq \mathbf{M}'(f)]]$.

The above corollary can be generalized by replacing $\mathbf{Ex}$ by $\mathbf{Lim}^n\mathbf{Ex}$ and $\mathbf{LimEx}$ by $\mathbf{Lim}^{n+1}\mathbf{Ex}$.

6 Strong Separation

We now prove the following theorem. A corollary (Corollary 24) of its proof provides a strong separation result. Other strong separation results can be found in [11, 8].

Theorem 23 $(\forall j \in N)(\exists \mathcal{C} \subseteq \mathcal{R})(\forall \mathbf{M})(\exists \mathbf{M}')(\forall a \in N \cup \{*\})[[\mathcal{C} \subseteq \mathbf{LimEx}(\mathbf{M}')] \wedge [\mathcal{C} \not\in \mathbf{Bc}^j] \wedge [(\mathbf{Ex}^a(\mathbf{M}) - \mathcal{C}) \subseteq \mathbf{Ex}^a(\mathbf{M}')] \wedge [(\mathbf{Bc}^a(\mathbf{M}) - \mathcal{C}) \subseteq \mathbf{Bc}^a(\mathbf{M}')]]$.

Proof. Fix j. Let $\mathbf{M}_0, \mathbf{M}_1, \ldots$ denote a standard recursive enumeration of all the inductive inference machines.

We will construct functions p and q such that, for each i, the following five conditions are satisfied.

(a) $\varphi^1_{p(i)} \in \mathcal{R}$.

(b) $\varphi^1_{p(i)} \not\in \mathbf{Bc}^j(\mathbf{M}_i)$.

(c) $\varphi^1_{p(i)}(0) = i$ and $\varphi^1_{p(i)}(1) = 0$.

(d)

(d.1) $\varphi_{q(i)} = \varphi^1_{p(i)}$ OR

(d.2) $\varphi_{q(i)}$ is an initial segment of $\varphi^1_{p(i)}$ and range$(\varphi^1_{p(i)} - \varphi_{q(i)}) = \{0\}$.

(e) $(\forall x > 0)[$

$[\varphi^1_{p(i)}(x) = 0 \wedge \varphi^1_{p(i)}(x+1) \neq 0] \Rightarrow$

$[(\forall r \mid 1 \leq r \leq j+1)[\varphi^1_{p(i)}(x+r) = 1] \wedge [\varphi^1_{p(i)}(x+j+2) = 2 + \Phi_{q(i)}(x+j+1)] \wedge [\varphi^1_{p(i)}(x+j+3) = 0]]$

$]$.

Let $\mathcal{C} = \{\varphi^1_{p(i)} \mid i \in N\}$.

We will define p, q as claimed above later. First we show, given $\mathbf{M}$, how to construct $\mathbf{M}'$ as claimed in the theorem (assuming the existence of p and q as defined above).

For $n > 0$, let $Sat(f[n])$ be true iff the following two conditions are satisfied.

(1) $(\forall x < n)[\Phi_{q(f(0))} < n \Rightarrow \varphi_{q(f(0))}(x) = f(x)]$.

(2) $(\forall x < n - j - 4)[$

$[f(x) = 0 \wedge f(x+1) \neq 0] \Rightarrow$

$[(\forall r \mid 1 \leq r \leq j+1)[f(x+r) = 1] \wedge [f(x+j+2) = 2 + \Phi_{q(f(0))}(x+j+1)] \wedge [f(x+j+3) = 0]]$

$]$.

It is easy to see that, for all f, $f \in \mathcal{C}$ iff $(\forall n > 0)[Sat(f[n])]$.

Let $\mathbf{M}'(f[0]) = 0$. For $n > 0$, if $Sat(f[n])$, then let $\mathbf{M}'(f[n]) = p(f(0))$; else let $\mathbf{M}'(f[n]) = \mathbf{M}(f[n])$. It is easy to see, from the properties of Sat discussed above that $\mathbf{M}'$ satisfies the conditions of the theorem.

We now construct p, q as claimed above. By, for example, the operator recursion theorem [6] there exist recursive functions p and q such that the (partial) functions $\varphi_{p(i)}$ and $\varphi_{q(i)}$ may be defined as follows. We define $\varphi_{p(i)}$ and $\varphi_{q(i)}$ in stages. Let x_s denote the least x such that $\varphi_{q(i)}$ is not defined before stage s.

Description of $\varphi_{p(i)}$ and $\varphi_{q(i)}$

```
For all $t \in N$, let $\varphi_{p(i)}(0,t) = i$. Let $\varphi_{q(i)}(0) = i$.
For all $t \in N$, let $\varphi_{p(i)}(1,t) = 0$. Let $\varphi_{q(i)}(1) = 0$.
Go to stage 0.
Begin stage $s$
1.    Dovetail steps 2 and 3 until step 2 succeeds. If and when step 2 succeeds,
        go to step 4.
2.    Let $g$ be the zero-extension of $\varphi_{q(i)}[x_s]$. Search for $n > x_s$, and for $w > n$,
        such that for each $r \leq j$, $\varphi_{\mathbf{M}_i(g[n])}(w+r)\downarrow = 0$.
3.    repeat
              Let $\langle x,t \rangle$ be the least number such that $\varphi_{p(i)}(x,t)$ has not been
                defined till now. Let $\varphi_{p(i)}(x,t) = 0$.
      forever
4.    Let $n, w$ be as found in step 2.
4.1   For each $x$, such that $x_s \leq x < w$
              Let $\varphi_{q(i)}(x) = 0$.
              For all $t$ such that $\varphi_{p(i)}(x,t)$ has not been defined till now, let
                $\varphi_{p(i)}(x,t) = 0$.
4.2   For each $r \leq j$, let $\varphi_{q(i)}(w+r) = 1$.
      For each $r \leq j$ and $t \in N$, such that $\varphi_{p(i)}(w+r,t)$ has not been defined
        till now, let $\varphi_{p(i)}(w+r,t) = 1$.
4.3   Let $\varphi_{q(i)}(w+j+1) = 2 + \Phi_{q(i)}(w+j)$.
      For all $t \in N$ such that $\varphi_{p(i)}(w+j+1,t)$ has not been defined till now,
        let $\varphi_{p(i)}(w+j+1,t) = 2 + \Phi_{q(i)}(w+j)$.
4.4   Let $\varphi_{q(i)}(w+j+2) = 0$.
      For all $t$ such that $\varphi_{p(i)}(w+j+2,t)$ has not been defined till now, let
        $\varphi_{p(i)}(w+j+2,t) = 0$.
```

5. Go to stage $s+1$.

End stage s

End of description of $\varphi_{p(i)}$ and $\varphi_{q(i)}$

Properties (a), (c), (d) and (e) follow immediately from construction. For property (b) consider the following cases.

Case 1: Infinitely many stages are executed.

In this case clearly, $\varphi^1_{p(i)} = \varphi_{q(i)} \in \mathcal{R}$. Moreover for infinitely many n, $\varphi_{\mathbf{M}_i(\varphi^1_{p(i)}[n])} \neq^j \varphi^1_{p(i)}$ (by the success of step 2, in each stage s and the diagonalization at step 4.2). Thus $\varphi^1_{p(i)} \not\in \mathbf{Bc}^j$.

Case 2: Some stage s starts but never finishes.

Let stage s be the stage which starts but never finishes. Clearly, $\varphi_{q(i)} = \varphi^1_{p(i)}[x_s]$ and $\varphi^1_{p(i)}$ is the zero-extension of $\varphi_{q(i)}[x_s]$. Moreover, for all $n > x_s$, for infinitely many x, $\varphi_{\mathbf{M}_i(\varphi^1_{p(i)}[n])}(x) \neq 0$ (otherwise, step 2 would succeed in stage s).

From the above cases it follows that p also satisfies condition (b). The theorem follows. ▮

Since $\lim_{n\to\infty} Sat(f[n])$, in the above proof, can be used to distinguish between functions in $\mathcal{C}$ and not in $\mathcal{C}$, the following corollary can be obtained by an easy modification of $\mathbf{M}'$ above.

Corollary 24 $(\forall a \in N \cup \{*\})(\forall \mathcal{C} \in \mathbf{Ex}^a)(\exists \mathcal{C}' \supseteq \mathcal{C})[\mathcal{C}' \in \mathbf{LimEx}^a - \mathbf{Ex}^a]$.

At present it is open whether the above theorem can be generalized to show that $(\forall a \in N \cup \{*\})(\forall n)(\forall \mathcal{C} \in \mathbf{Lim}^n\mathbf{Ex}^a)(\exists \mathcal{C}' \supseteq \mathcal{C})[\mathcal{C}' \in \mathbf{Lim}^{n+1}\mathbf{Ex}^a - \mathbf{Lim}^n\mathbf{Ex}^a]$.

7 Extension of LimEx Criterion to Learning with Additional Information

7.1 Learning with Additional Information

In [17, 22] learning with the knowledge of upper bound on the program size of the function being learned was introduced. In this approach, a machine identifying a program for a recursive function from its graph, is provided with an upper-bound on the minimal program for the function as additional information. Hence, to simplify our exposition, learning machines with additional information can be viewed as taking two arguments: an upper-bound on the minimal program and an initial segment of graph of a function. In other words, learning machines identifying functions with additional information are algorithmic devices that compute a mapping from $N \times \mathrm{INIT}$ into N.

$\mathbf{M}(j,f)\downarrow$ (read: $\mathbf{M}$ on f with additional information j converges) iff $(\exists i)(\overset{\infty}{\forall} n)[\mathbf{M}(j,f[n]) = i]$. $\mathbf{M}(j,f)\uparrow$ otherwise. If $\mathbf{M}(j,f)\downarrow$, then $\mathbf{M}(j,f) = i$, where i is such that $(\overset{\infty}{\forall} n)[\mathbf{M}(j,f[n]) = i]$.

Definition 25 [22] Let $a,b \in N \cup \{*\}$.

(a) $\mathbf{M}$ $\mathbf{Bex}^{a,b}$-*identifies* $f \in \mathcal{R}$ (written: $f \in \mathbf{Bex}^{a,b}(\mathbf{M})$) $\iff$ $(\forall x \geq \mathrm{MinProg}^{b}(f))$ $(\exists i \mid \varphi_i =^{a} f)$ $[\mathbf{M}(x,f)\downarrow = i]$.

(b) $\mathbf{Bex}^{a,b} = \{\mathcal{C} \subseteq \mathcal{R} \mid (\exists \mathbf{M})[\mathcal{C} \subseteq \mathbf{Bex}^{a,b}(\mathbf{M})]\}$.

Intuitively, $\mathbf{M}$ $\mathbf{Bex}^{a,b}$-identifies f iff $\mathbf{M}$, fed j, at least as large as the minimal program that computes f with at most b errors, and graph of f, converges to a program that computes f with at most a errors. The notion $\mathbf{Bex}^{0,0}$-identification was first studied by Freivalds and Wiehagen [17].

If, in Definition 25, we further require that the final program conjectured be the same for any upper-bound, we get a new identification criterion described in Definition 26 below.

Definition 26 [22] Let $a,b \in N \cup \{*\}$.

(a) $\mathbf{M}$ $\mathbf{ResBex}^{a,b}$-*identifies* $f \in \mathcal{R}$ (written: $f \in \mathbf{ResBex}^{a,b}(\mathbf{M})$) $\iff$ $(\exists i \mid \varphi_i =^{a} f)$ $(\forall x \geq \mathrm{MinProg}^{b}(f))$ $[\mathbf{M}(x,f)\downarrow = i]$.

(b) $\mathbf{ResBex}^{a,b} = \{\mathcal{C} \subseteq \mathcal{R} \mid (\exists \mathbf{M})[\mathcal{C} \subseteq \mathbf{ResBex}^{a,b}(\mathbf{M})]\}$.

$\mathbf{ResBex}^{0,0}$-identification criterion was mentioned as GN^{+} in [16].

It is noted in [16, 22] that $\mathbf{ResBex}^{0,0} \subset \mathbf{Bex}^{0,0}$. In [22] this is interpreted to indicate a possible explanation for the superiority of non-classroom over classroom language learning: in the classroom one is expected to learn the same grammar as the teacher. It is proved in [22] that for all $n \in N$, $\mathcal{R} \in \mathbf{Bex}^{n,n}$.

7.2 Learning Limiting Programs with Additional Information

We now define the notion of learning limiting program with additional information.

Definition 27 Let $a,b \in N \cup \{*\}$.

(a) $\mathbf{M}$ $\mathbf{BLimEx}^{a,b}$-*identifies* $f \in \mathcal{R}$ (written: $f \in \mathbf{BLimEx}^{a,b}(\mathbf{M})$) $\iff$ $(\forall x \geq \mathrm{MinProg}^{b}(f))$ $(\exists i \mid \varphi^{1}_i =^{a} f)$ $[\mathbf{M}(x,f)\downarrow = i]$.

(b) $\mathbf{BLimEx}^{a,b} = \{\mathcal{C} \subseteq \mathcal{R} \mid (\exists \mathbf{M})[\mathcal{C} \subseteq \mathbf{BLimEx}^{a,b}(\mathbf{M})]\}$.

Definition 28 Let $a, b \in N \cup \{*\}$.

(a) $\mathbf{M}$ $\mathbf{ResBLimEx}^{a,b}$-*identifies* $f \in \mathcal{R}$ (written: $f \in \mathbf{ResBLimEx}^{a,b}(\mathbf{M})$) $\iff$ $(\exists i \mid \varphi_i^1 =^a f)$ $(\forall x \geq \mathrm{MinProg}^b(f))$ $[\mathbf{M}(x, f)\downarrow = i]$.

(b) $\mathbf{ResBLimEx}^{a,b} = \{\mathcal{C} \subseteq \mathcal{R} \mid (\exists \mathbf{M})[\mathcal{C} \subseteq \mathbf{ResBLimEx}^{a,b}(\mathbf{M})]\}$.

Theorem 29 $\mathcal{R} \in \mathbf{ResBLimEx}^{0,0}$.

PROOF. Consider the following function:

$F(0) = 0$. $F(i) = \min(N - \{F(j) \mid j < i \wedge (\exists x)[\varphi_j(x)\downarrow \neq \varphi_i(x)\downarrow]\})$.

For each i define G_i as follows.

Given i, x, let $j = \min(\{k \mid F(k) = i \wedge \varphi_k(x)\downarrow\})$ and then set $G_i(x) = \varphi_j(x)$.

It is easy to see that both F and G_i are total and that φ^1-programs for them can be found algorithmically from i. Also for all i, j such that $\varphi_i = \varphi_j \in \mathcal{R}$: $F(i) = F(j)$ and $G_{F(i)} = \varphi_i$. The theorem follows from the fact that $\mathcal{R} \in \mathbf{Bex}^{0,0}$ [17, 22]. ▮

Proof techniques used in [22] can be used to prove several more theorems regarding the relationships between the criteria of this section.

References

[1] D. Angluin and C. Smith. Inductive inference: Theory and methods. *Computing Surveys*, 15:237–289, 1983.

[2] G. Baliga, J. Case, S. Jain, and M. Suraj. Machine learning of higher order programs. In *Proceedings of the Second Symposium on Logical Foundations of Computer Science, Tver, Russia.*, volume 620 of *Lecture Notes in Computer Science*, pages 9–20. Springer-Verlag, 1992.

[3] 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.

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

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

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

[7] J. Case. Learning machines. In W. Demopoulos and A. Marras, editors, *Language Learning and Concept Acquisition*, pages 83–102. Ablex Publishing Company, 1986.

[8] J. Case, K. J. Chen, and S. Jain. Strong separation of learning classes. *Journal of Experimental and Theoretical Artificial Intelligence*, 4(4):281–293, 1992.

[9] J. Case, S. Jain, and A. Sharma. Anomalous learning helps succinctness. In S. Arikawa, S. Goto, S. Ogsuga, and T. Yokomori, editors, *Proceedings of the First Workshop on Algorithmic Learning Theory*, pages 282–288. Japanese Society for Artificial Intelligence, 1990.

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

[11] K. J. Chen. *Tradeoffs in Machine Inductive Inference.* PhD thesis, SUNY/Buffalo, 1981.

[12] K. J. Chen. Tradeoffs in inductive inference of nearly minimal sized programs. *Information and Control*, 52:68–86, 1982.

[13] W. Craig. On axiomatizability within a system. *Journal of Symbolic Logic*, 18:30–32, 1953.

[14] S. Feferman. Arithmetization of metamathematics in a general setting. *Fundamenta Mathematicae*, 49:35–92, 1960.

[15] R. Freivalds. Minimal Gödel numbers and their identification in the limit. In *Mathematical Foundations of Computer Science*, volume 32 of *Lecture Notes in Computer Science*, pages 219–225. Springer-Verlag, 1975.

[16] R. Freivalds, E. Kinber, and R. Wiehagen. Connections between identifying functionals, standardizing operations, and computable numberings. *Zeitschr. j. math. Logik und Grundlagen d. Math. Bd.*, 30:145–164, 1984.

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

[18] M. Fulk. *A Study of Inductive Inference Machines.* PhD thesis, SUNY/Buffalo, 1985.

[19] E. M. Gold. Limiting recursion. *Journal of Symbolic Logic*, 30:28–48, 1965.

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

[21] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979.

[22] S. Jain and A. Sharma. Learning with the knowledge of an upper bound on program size. *Information and Computation*, 102:118–166, 1993.

[23] R. Klette and R. Wiehagen. Research in the theory of inductive inference by GDR mathematicians – A survey. *Information Sciences*, 22:149–169, 1980.

[24] N. Lynch, A. Meyer, and M. Fischer. Relativization of the theory of computational complexity. *Transactions of the American Mathematical Society*, 220:243–287, 1976.

[25] M. Machtey and P. Young. *An Introduction to the General Theory of Algorithms.* North Holland, New York, 1978.

[26] Y. Marcoux. Composition is almost as good as s-1-1. In *Proceedings, Structure in Complexity Theory–Fourth Annual Conference.* IEEE Computer Society Press, 1989.

[27] E. Mendelson. *Introduction to Mathematical Logic.* Brooks-Cole, San Francisco, 1986. 3rd Edition.

[28] D. Osherson, M. Stob, and S. Weinstein. *Systems that Learn: An Introduction to Learning Theory for Cognitive and Computer Scientists.* MIT Press, 1986.

[29] H. Putnam. Trial and error predicates and the solution to a problem of Mostowski. *Journal of Symbolic Logic*, 30:49–57, 1965.

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

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

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

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

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

[35] J. Royer and J. Case. *Intensional Subrecursion and Complexity Theory.* Research Notes in Theoretical Science. Pitman Press, being revised for expected publication, 1992.

[36] N. Shapiro. Review of "Limiting recursion" by E.M. Gold and "Trial and error predicates and the solution to a problem of Mostowski" by H. Putnam. *Journal of Symbolic Logic*, 36:342, 1971.

[37] J. Shoenfield. On degrees of unsolvability. *Annals of Mathematics*, 69:644–653, 1959.

[38] J. Shoenfield. *Degrees of Unsolvability.* North-Holland, 1971.

[39] R. Soare. *Recursively Enumerable Sets and Degrees.* Springer-Verlag, 1987.

[40] R. Wiehagen. *Zur Theorie der Algorithmischen Erkennung.* PhD thesis, Humboldt University of Berlin, 1978.