Complexity Issues for Vacillatory Function Identification [1]

John Case
Department of Computer and Information Sciences
University of Delaware
Newark, DE 19716, USA
Email: case@cis.udel.edu

Sanjay Jain
Institute of Systems Science
National University of Singapore
Singapore 0511
Republic of Singapore
Email: sanjay@iss.nus.sg

Arun Sharma
School of Computer Science and Engineering
The University of New South Wales
Sydney, NSW 2033, Australia
Email: arun@spectrum.cs.unsw.oz.au

[1]Work supported by NSF grant CCR 871-3846 and a grant from Siemen's Corporation to MIT. Preliminary version of this paper appeared in the Proceedings of the Foundations of Software Technology and Theoretical Computer Science, Eleventh Conference, New Delhi, India, December 1991. Part of the work was done while the second author was at the University of Delaware and the third author was at the Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology, Cambridge, MA, USA.

Abstract

It was previously shown by Barzdin and Podnieks that one does *not* increase the power of learning programs for *functions* by allowing learning algorithms to converge to a finite set of correct programs instead of requiring them to converge to a single correct program. In this paper we define some new, subtle, but natural concepts of mind change complexity for function learning and show that, if one bounds this complexity for learning algorithms, then, by contrast with Barzdin and Podnieks result, there are interesting and sometimes complicated tradeoffs between these complexity bounds, bounds on the number of final correct programs, and learning power.

CR Classification Number: I.2.6 (Learning – Induction).

1 Introduction

Let $N = \{0, 1, 2, \ldots\}$, the set of natural numbers. Let f be any function : $N \rightarrow N$. For any $n \in N$, we let $f[n]$ denote $\{(x, f(x)) \mid x < n\}$, the finite initial segment of f consisting of the first n data points in the graph of f. $\in$, $\subseteq$, and $\subset$ denote, respectively, membership, containment, and proper containment for sets. The quantifier '$\overset{\infty}{\forall}$' means 'for all but finitely many natural numbers'.

A *function learning machine* $\mathbf{M}$ is an algorithmic device which, on any input $f[n]$, returns either ? or a program, where, if it returns a program on a segment, it returns a program on all extensions of that segment. We write $\mathbf{M}(f[n])$ for the output of $\mathbf{M}$ on $f[n]$. If $\mathbf{M}(f[n])$ is a program, we think of that program as $\mathbf{M}$'s conjecture, based on the data $f[n]$, as to how to compute all of f; $\mathbf{M}(f[n]) =?$ then represents the situation where $\mathbf{M}$ does not conjecture a program based on the data $f[n]$. The restriction that $\mathbf{M}$ must continue to conjecture programs once it has done so is essentially without loss of generality since, intuitively, a machine which hasn't had enough time to think of a new conjecture can be thought of re-outputting its previous conjecture.

As is by now well known, there are various senses in which $\mathbf{M}$ can be thought of as *successfully* learning or inferring a program for f. Let $o_i = \mathbf{M}(f[i])$. The criterion of success known as **Ex**-*identification* (Gold (1967), Blum and Blum (1975), and Case and Smith (1983)) requires that the sequence $o_0, o_1, o_2, \ldots$ contains a *program* p *for* f such that $(\overset{\infty}{\forall} i)[o_i = p]$. In this case one speaks of p as being the *final* program output by $\mathbf{M}$ on f. Barzdin and Podnieks (1973) (see also Case and Smith (1983)) considered a vacillatory criterion of success (called **Fex**-*identification* in Case and Smith (1983)) which requires that on f but *finitely* many distinct programs are output and that the set P of *programs* appearing in $o_0, o_1, o_2, \ldots$ infinitely often be non-empty (and finite) and, furthermore, that each program in P computes f. In this case P is spoken of as the *set* of final program*s* output by $\mathbf{M}$ on f. It was shown by Barzdin and Podnieks (1973) that, in spite of **Fex** being a pleasantly more liberal criterion of success, one can translate any function learning machine **Fex**-identifying a class of functions into one **Ex**-identifying the same class of functions.[1] An examination of the translation given by Case and Smith (1983) shows that the function learning machine output by the translation will ostensibly take more "time" to reach its final program than the input function learning machine does to reach its final program*s*.

In this paper we show, then, that there are some interesting effects on learning power, *for vacillatory function learning*, wrought by bounding *suitably sensitive* measures of the computational complexity of the learning machines themselves. Our suitably sensitive measures need to be pleasingly more subtle than ordinary mind change complexity (Barzdin and Freivalds (1974), Case and Smith (1983)) (Proposition 1); however, our measures are much closer to ordinary mind change complexity than to the more comprehensive complexity measures in Wiehagen (1986), Angluin (1980), Gold (1978), Daley and Smith (1986). It intriguingly remains to be investigated whether there is also an effect on learning power that also occurs for more comprehensive measures.

Suppose $o_0, o_1, o_2, \ldots$ is the sequence of outputs of $\mathbf{M}$ on f. The first, not very subtle, complexity measure one might think of for *vacillatory* function learning is to count the number of changes of conjecture beyond the first program in $o_0, o_1, o_2, \ldots$ up until nothing but the *final* successful programs appear in $o_0, o_1, o_2, \ldots$. It is very straightforward to show (Proposition 10) that vacillatory function identification subjected to a bound on *this* complexity measure yields no more learning power than bounding the number of mind changes for **Ex**-identification. Hence, we will look to somewhat more sophisticated measures of complexity: just as the custom of pressing a flower in a book teases apart the petals, so does a harder conceptual perspective tease apart concepts that weren't apart from a weaker perspective.[2] As will be seen, our perspectives on mind change-like complexity reveal a deeper structure than the usual perspectives.

Suppose we are given an *a priori* bound b on the cardinality of the set of final programs. Suppose, *for example*, that $b = 2$ and $\mathbf{M}$ on f outputs the sequence

$$?, p_1, p_2, p_1, p_3, p_1, p_2, p_1, p_4, p_3, p_4, \ldots, \tag{A}$$

where the p_i's are all programs and the continuation of the sequence is an infinite alternation between p_3 and p_4. Suppose, with an eye to eventually "finding" the non-empty, finite set of *final* programs P output (if it exists), at each new output of $\mathbf{M}$ on f we *non-deterministically* choose a candidate for P ("rationally" based on the value of b and the outputs so far). For example, based on $b = 2$ and the sequence (A), we *might* choose the succession of candidates

$$\emptyset, \{p_1\}, \{p_1, p_2\}, \{p_1, p_2\}, \{p_2, p_3\}, \{p_1, p_3\}, \{p_1, p_2\}, \{p_1, p_2\}, \{p_4\}, \{p_3, p_4\}, \{p_3, p_4\}, \ldots, \tag{B}$$

[1]Interestingly, in the context of machine learning of grammars for languages, vacillatory criteria *do* yield more learning power (Osherson and Weinstein (1982)). In fact in *this* context increasing by one a constant bound on the *number* of final grammars leads to an increase in learning power (Case (1988)). See also Case, Jain and Sharma (1989).

[2]This metaphor was suggested by Mark Fulk.

where the sequence continues with endless repetitions of $\{p_3, p_4\}$. This choice is rational because no candidate has cardinality exceeding 2 and, *if* a new candidate is chosen differing from the just previous candidate, the only *addition* is the latest *program* conjectured; furthermore, each choice *contains* the current output program (if any). If, as in this example, the sequence of sets chosen stabilizes to the actual (non-empty, finite) final set of programs, we measure the number of *set mind changes* as the number of changes of candidate chosen beyond the first non-empty candidate. For the choice sequence (B), the number of set mind changes is 6. A different "rational" choice sequence of candidates for the output sequence (A), might yield a different number of set mind changes. For example,

$$\emptyset, \{p_1\}, \{p_1, p_2\}, \{p_1, p_2\}, \{p_1, p_3\}, \{p_1, p_3\}, \{p_1, p_2\}, \{p_1, p_2\}, \{p_4\}, \{p_3, p_4\}, \{p_3, p_4\}, \qquad \text{(C)}$$

where the sequence continues with endless repetitions of $\{p_3, p_4\}$, is also rational, but involves but 5 set mind changes. We measure the set mind change complexity by the *smallest* number of set mind changes available among the rational, non-deterministic choices. For the value of $b = 2$ and output sequence (A), 5 is the measure of set mind change complexity.

$\mathbf{Sfex}^a_{b,c}$-identification, introduced formally below in Definitions 11 and 12, is a vacillatory function identification criterion requiring that

(a) the cardinality of the non-empty, finite set of final programs be $\leq b$,

(b) the set mind change complexity is $\leq c$, and

(c) each of the final programs computes the input function with mistakes (anomalies) at $\leq a$ arguments.

$\mathbf{Sfex}^a_{b,c}$ is the class of sets $\mathcal{C}$ of functions such that some function learning machine $\mathbf{Sfex}^a_{b,c}$-identifies $\mathcal{C}$. We usually drop the superscript in $\mathbf{Sfex}^a_{b,c}$ (and $\mathbf{Qfex}^a_{b,c}$ defined below) when the superscript is 0.

By Corollary 42 below we have that

$$\mathbf{Sfex}_{1,c} \subset \mathbf{Sfex}_{2,c} \subset \ldots \subset \mathbf{Sfex}_{c,c} = \mathbf{Sfex}_{c+1,c} = \ldots. \qquad \text{(D)}$$

Therefore, in the context of function learning, for a constant set mind change bound, increasing the bound on the size of the final set of programs (up to the set mind change bound) strictly increases learning power.

For $c > 1$, Theorems 16 and 18 below imply, for example, that

$$\mathbf{Sfex}_{2,c} \subset \mathbf{Sfex}_{1,2c-1}. \qquad \text{(E)}$$

Hence, from (D) there is a set of functions $\mathcal{C}$ which, for a set mind change bound of c, requires a vacillation bound of at least 2 on the cardinality of the set of final programs, but by (E), one can reduce the vacillation bound to 1 by increasing the set mind change bound to $2c-1$. Therefore, there are cases *in the context of function learning* in which an increase in the vacillation bound results in a strict set mind change complexity savings!

A special case of our hardest simulation result (Theorem 18 below) yields that[3]

$$\mathbf{Sfex}_{3,3} \subseteq \mathbf{Sfex}_{2,4}. \tag{F}$$

Hence, an increase in set mind change complexity (from 3 to 4) can in all cases allow a decrement in vacillation bound (from 3 to 2). The most general tradeoff formula of this sort we have so far, in Theorem 18, is fairly complex.

We also define a variant of $\mathbf{Sfex}^a_{b,c}$-identification, called $\mathbf{Qfex}^a_{b,c}$-*identification* (Definitions 13 and 14), in which the non-deterministic choices for the non-empty, finite *set* of final programs P are naturally replaced by choices of *queues* whose corresponding sets of elements are candidates for P. Complexity tradeoff results for $\mathbf{Qfex}^a_{b,c}$-identification hold which are quite similar to those for $\mathbf{Sfex}^a_{b,c}$-identification. Many of our theorems in Sections 3.2 and 3.3 specify what we know of the complicated relationship between these set and queue based mind change vacillatory function learning criteria.

It is also interesting to consider the complexity of final programs. In the prior literature there are two basic measures of the complexity of final programs:

(a) program size (Kinber (1977), Kinber (1974), Freivalds (1975), Chen (1981), Chen (1982), Jain and Sharma (1990), Case, Jain and Sharma (1989)) and

(b) program speed (Sipser cited in Chen (1981) and Zeugmann (1983)).

We consider the first case and answer an open question in Chen (1981) (Theorem 48 below) for **Mfex**-*identification*, i.e., **Fex**-identification with no bound on the finite size of the non-empty set of final programs but with the requirement that the final programs be of size within some pre-assigned computable function of minimal size. **Mex**-*identification* is the special case of **Mfex**-identification in which the set of final programs is required to be a singleton. The result is that there is a translation of any function learning machine **Mfex**-identifying a class of functions into one **Mex**-identifying the same class of functions.

2 Preliminaries

Recursion-theoretic concepts not explained below are treated in Rogers (1967). N^+ denotes the set of positive integers, $\{1, 2, 3, \ldots\}$.

$*$ denotes a non-member of N and is assumed to satisfy $(\forall n)[n < * < \infty]$. d, e, i, j, k, l, m, n, o, q, r, s, t, w, x, y, z, with or without decorations range over N. a, b, c, with or without decorations range over $N \cup \{*\}$. In some contexts, p, with or without decorations, ranges over N, being construed as program for a (partial) function. In other contexts,

[3]By Theorem 16 the inclusion in (F) is actually proper.

p, ranges over total function, with the range of p being construed as programs for (partial) functions.

We let P, S, with or without decorations, range over subsets of N and we let D range over finite subsets of N. D_x denotes the finite set with canonical index x (Rogers (1967)). We often identify finite sets with their canonical indices. This is usually done for notational convenience when (canonical indices of) finite sets are input parameters for programs. $\langle \cdot, \cdot \rangle$ denotes a 1-1 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. $\text{card}(P)$ denotes the cardinality of P. So then, '$\text{card}(P) \leq *$' means that $\text{card}(P)$ is finite. $\min(P)$ and $\max(P)$ respectively denote the minimum and maximum element in P. We take $\min(\emptyset)$ to be ∞ and $\max(\emptyset)$ to be 0.

η, with or without decorations, ranges over partial functions. For $a \in (N \cup \{*\})$, $\eta_1 =^a \eta_2$ means that $\text{card}(\{x \mid \eta_1(x) \neq \eta_2(x)\}) \leq a$. $\text{domain}(\eta)$ and $\text{range}(\eta)$ respectively denote the domain and range of partial function η. In some contexts s, with or without decorations, ranges over finite sequences. For finite sequences s_1 and s_2, $s_1 \diamond s_2$ denotes the concatenation of s_1 and s_2.

$\mathcal{R}$ denotes the class of all *recursive* functions, i.e., total computable functions with arguments and values from N. f, g, h, with or without decorations, range over $\mathcal{R}$. $\mathcal{C}$ and $\mathcal{S}$, with or without decorations, range over subsets of $\mathcal{R}$.

φ denotes a fixed *acceptable* programming system for the partial computable functions: $N \rightarrow N$ (Rogers (1958), Rogers (1967), Machtey and Young (1978)). (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 (Riccardi (1980), Riccardi (1981), Royer (1987), Marcoux (1989).) φ_i denotes the partial computable function computed by program i in the φ-system. We let Φ be an arbitrary Blum complexity measure (Blum (1967)) associated with acceptable programming system φ; such measures exist for any acceptable programming system (Blum (1967)). We let *pad* be a 1-1 total function such that $(\forall i, j)[\varphi_{pad(\langle i,j \rangle)} = \varphi_i]$. For a given total computable function f, we define $\text{MinProg}(f)$ to denote $\min(\{i \mid \varphi_i = f\})$. For an integer u, $ABS(u)$ denotes the absolute value of u.

The quantifiers '$\overset{\infty}{\exists}$' and '$\exists!$' mean 'there exist infinitely many,' and 'there exists a unique' respectively.

Hereinafter we refer to function learning machines as learning machines. SEG denotes the set of all finite initial segments. We let σ, τ, γ and ξ, with or without decorations, range over SEG. We let $\mathbf{M}$, with or without decorations, range over learning machines.

In Definition 1 below we spell out what it means for a learning machine on a function to converge in the limit.

Definition 1 Suppose $\mathbf{M}$ is a learning machine and f is a computable function. $\mathbf{M}(f)\downarrow$ (read: $\mathbf{M}(f)$ *converges*) just in case $(\exists i)(\overset{\infty}{\forall} n)$ $[\mathbf{M}(f[n]) = i]$. If $\mathbf{M}(f)\downarrow$, then $\mathbf{M}(f)$ is defined = the unique i such that $(\overset{\infty}{\forall} n)[\mathbf{M}(f[n]) = i]$, otherwise we say that $\mathbf{M}(f)$ diverges (written: $\mathbf{M}(f)\uparrow$).

We now introduce a criteria for a learning machine to successfully infer a function.

Definition 2 (Gold (1967), Blum and Blum (1975), Case and Smith (1983)) Let $a \in N \cup \{*\}$.
(i) $\mathbf{M}$ $\mathbf{Ex}^a$-*identifies* f (written: $f \in \mathbf{Ex}^a(\mathbf{M})$) just in case $(\exists i \mid \varphi_i =^a f)[\mathbf{M}(f)\downarrow = i]$.
(ii) $\mathbf{Ex}^a = \{\mathcal{S} \mid (\exists \mathbf{M})[\mathcal{S} \subseteq \mathbf{Ex}^a(\mathbf{M})]\}$.

Note that $\mathbf{Ex}^0$ is the same as the class $\mathbf{Ex}$ defined in the introduction.

Theorem 3 *For all $a \in N$,*
(a) (Case and Smith (1983)) $\mathbf{Ex}^a \subset \mathbf{Ex}^{a+1}$.
(b) (Case and Smith (1983)) $\bigcup_{a\in N} \mathbf{Ex}^a \subset \mathbf{Ex}^*$.
(c) $\mathcal{R} \notin \mathbf{Ex}^*$.

The notion of $\mathbf{Ex}^*$ identification is due to Blum and Blum (1975). Gold's (Gold (1967)) proof for $\mathcal{R} \notin \mathbf{Ex}$ also shows that $\mathcal{R} \notin \mathbf{Ex}^*$. Case and Smith (1983) (see also Barzdin and Freivalds (1974)) introduce a refinement of the above notion of $\mathbf{Ex}$-identification by bounding the number of times a learning machine is allowed to change its mind before converging to a correct program for the function being learned. Definition 4 below describes this notion.

Definition 4 (Case and Smith (1983)) Suppose $a, c \in N \cup \{*\}$.
(i) $\mathbf{M}$ $\mathbf{Ex}^a_c$-*identifies* f (written: $f \in \mathbf{Ex}^a_c(\mathbf{M})$) just in case $[(\exists i \mid \varphi_i =^a f)(\overset{\infty}{\forall} n)[\mathbf{M}(f[n]) = i] \wedge \mathrm{card}(\{n \mid ? \neq \mathbf{M}(f[n]) \neq \mathbf{M}(f[n+1])\}) \leq c]$.
(ii) $\mathbf{Ex}^a_c = \{\mathcal{C} \mid (\exists \mathbf{M})[\mathcal{C} \subseteq \mathbf{Ex}^a_c(\mathbf{M})]\}$.

Clearly, Gold's notion of $\mathbf{Ex}$-identification is the same as $\mathbf{Ex}^0_*$-identification. We usually drop the superscript in $\mathbf{Ex}^a_b$ when the superscript is 0.

In Definition 5 just below we spell out what it means for a learning machine on a function to converge in the limit to a finite set of programs.

Definition 5 Suppose $\mathbf{M}$ is a learning machine and $f \in \mathcal{R}$. $\mathbf{M}(f)\Downarrow$ (read: $\mathbf{M}(f)$ *finitely-converges*) just in case $\{\mathbf{M}(f[n]) \mid n \in N\}$ is finite. If $\mathbf{M}(f)\Downarrow$, then we say that $\mathbf{M}(f)\Downarrow = D$, where $D = \{i \mid (\overset{\infty}{\exists} n)[\mathbf{M}(f[n]) = i]\}$; otherwise, for no D is $\mathbf{M}(f)\Downarrow = D$.

Definition 6 Suppose $a \in N \cup \{*\}$ and $b \in N^+ \cup \{*\}$.
(i) A learning machine, $\mathbf{M}$, is said to $\mathbf{Fex}^a_b$-*identify* a function f (written: $f \in \mathbf{Fex}^a_b(\mathbf{M})$) just in case $(\exists D \mid \mathrm{card}(D) \leq b)[\mathbf{M}(f)\Downarrow = D \wedge (\forall i \in D)[\varphi_i =^a f]]$.
(ii) $\mathbf{Fex}^a_b = \{\mathcal{C} \mid (\exists \mathbf{M})[\mathcal{C} \subseteq \mathbf{Fex}^a_b(\mathbf{M})]\}$.

In $\mathbf{Fex}_b^a$-identification, the b is a "bound" on the number of final programs and the a is a "bound" on the number of anomalies allowed in these final programs. A "bound" of $*$ just means unbounded, but finite. We usually drop the superscript in $\mathbf{Fex}_b^a$ when the superscript is 0.

Following result due to Case and Smith (1983) says that vacillatory function identification *does not* give any extra inferring power over $\mathbf{Ex}$-identification. In the following theorem $a = 0$ case was done by Barzdin and Podnieks (1973).

Theorem 7 (Case and Smith (1983)) $(\forall a \in N)[\mathbf{Fex}_*^a = \mathbf{Ex}^a]$.

3 Mind Changes for Vacillatory Function Identification

3.1 Definitions

We now formally define criteria for vacillatory identification with mind changes, which were informally discussed in the introduction.

Definition 8 Let $b \in N^+ \cup \{*\}$. Then n is a $\mathbf{Fex}_b$-*stabilizing point for* $\mathbf{M}$ *on* f just in case the following hold:
(1) $\mathrm{card}(\{\mathbf{M}(f[m]) \mid m \geq n\}) \leq b$.
(2) $(\forall m \geq n)(\overset{\infty}{\exists}\, i)[\mathbf{M}(f[m]) = \mathbf{M}(f[i])]$.

Definition 9 Let $a, c \in N \cup \{*\}$. Let $b \in N^+ \cup \{*\}$.
(1) A machine $\mathbf{M}$ $\mathbf{Fex}_{b,c}^a$-*identifies* f (written: $f \in \mathbf{Fex}_{b,c}^a(\mathbf{M})$) just in case $\mathbf{M}$ $\mathbf{Fex}_b^a$-identifies f and $\mathrm{card}(\{i < n \mid ? \neq \mathbf{M}(f[i]) \neq \mathbf{M}(f[i+1])\}) \leq c$, where n is the least $\mathbf{Fex}_b$ stabilizing point for $\mathbf{M}$ on f.
(2) $\mathbf{Fex}_{b,c}^a = \{\mathcal{C} \mid (\exists \mathbf{M})[\mathcal{C} \subseteq \mathbf{Fex}_{b,c}^a(\mathbf{M})]\}$.

Proposition 10 *For all* a, c, $\mathbf{Fex}_{*,c}^a = \mathbf{Ex}_c^a$.

We now formally define $\mathbf{Sfex}_{b,c}^a$-identification.

Definition 11 Let $b \in N^+ \cup \{*\}$. A sequence of sets $S_0, S_1, \ldots$ is $\mathbf{Fex}_b$-*valid for* $\mathbf{M}$ *on* f just in case, for all $i \in N$, the following five conditions hold.
(1) $\mathrm{card}(S_i) \leq b$.
(2) $S_0 = \{\mathbf{M}(f[0])\} - \{?\}$.
(3) $S_{i+1} - S_i \subseteq \{\mathbf{M}(f[i+1])\}$.
(4) $\mathbf{M}(f[i]) \in S_i \cup \{?\}$.
(5) $\lim_{j \to \infty} S_j$ exists and is $= \{p \mid (\overset{\infty}{\exists}\, k)[\mathbf{M}(f[k]) = p]\}$.

Definition 12 Let $a, c \in N \cup \{*\}$. Let $b \in N^+ \cup \{*\}$.
(1) A machine $\mathbf{M}$ $\mathbf{Sfex}^a_{b,c}$-*identifies* f (written: $f \in \mathbf{Sfex}^a_{b,c}(\mathbf{M})$) just in case $\mathbf{M}$ $\mathbf{Fex}^a_b$-identifies f and there exists a $\mathbf{Fex}_b$-valid sequence of sets $S_0, S_1, \ldots$ for $\mathbf{M}$ on f such that $\mathrm{card}(\{i \mid \emptyset \neq S_i \neq S_{i+1}\}) \leq c$.

We usually call, $\min(\{\mathrm{card}(\{i \mid \emptyset \neq S_i \neq S_{i+1}\}) \mid S_0, S_1, \ldots$ is an $\mathbf{Fex}_b$-valid sequence for $\mathbf{M}$ on $f\})$, as the number of *set*$_b$ *mind changes* by $\mathbf{M}$ on f; we often drop b from set_b, if b is clear from context.
(2) $\mathbf{Sfex}^a_{b,c} = \{\mathcal{C} \mid (\exists \mathbf{M})[\mathcal{C} \subseteq \mathbf{Sfex}^a_{b,c}(\mathbf{M})]\}$.

We now formally define **Qfex**-identification.

A *queue* is an ordered sequence of elements (not necessarily distinct). Let $Q = q_0 q_1 \ldots q_{n-1}$ be a queue of length n. q_0 is called the front of the queue. ( ) denotes the empty queue. $Q \diamond q$ is the queue $q_0 q_1 \ldots q_{n-1} q$. A tail of a queue is a queue obtained by deleting some (possibly none) of the elements from the front of the queue. Thus ( ) and $q_i q_{i+1} \ldots q_{n-1}$, $i < n$ are all tails of the queue Q. $\mathrm{Set}(Q)$ is the set of elements in the queue Q.

Definition 13 Let $b \in N^+ \cup \{*\}$. A sequence of queues $Q_0, Q_1, \ldots$ is $\mathbf{Fex}_b$-*valid for* $\mathbf{M}$ *on* f just in case, for all $i \in N$, the following five conditions hold.
(1) $\mathrm{card}(Q_i) \leq b$.
(2) $Q_0 = (\ )$ or $Q_0 = (\mathbf{M}(f[0]))$.
(3) Q_{i+1} is a tail of either Q_i or $Q_i \diamond (\mathbf{M}(f[i+1]))$.
(4) $\mathbf{M}(f[i]) \in \mathrm{Set}(Q_i) \vee \mathbf{M}(f[i]) =?$.
(5) $\lim_{j \to \infty} Q_j$ exists and $\mathrm{Set}(\lim_{j \to \infty} Q_j) = \{p \mid (\overset{\infty}{\exists} k)[\mathbf{M}(f[k]) = p]\}$.

Definition 14 Let $a, c \in N \cup \{*\}$. Let $b \in N^+ \cup \{*\}$.
(1) A machine $\mathbf{M}$ $\mathbf{Qfex}^a_{b,c}$-*identifies* f (written: $f \in \mathbf{Qfex}^a_{b,c}(\mathbf{M})$) just in case $\mathbf{M}$ $\mathbf{Fex}^a_b$-identifies f and there exists a $\mathbf{Fex}_b$-valid sequence of queues $Q_0, Q_1, \ldots$ for $\mathbf{M}$ on f such that $\mathrm{card}(\{i \mid (\ ) \neq Q_i \neq Q_{i+1}\}) \leq c$.

We usually call, $\min(\{\mathrm{card}(\{i \mid (\ ) \neq Q_i \neq Q_{i+1}\}) \mid Q_0, Q_1, \ldots$ is an $\mathbf{Fex}_b$-valid sequence for $\mathbf{M}$ on $f\})$, as the number of *queue*$_b$ *mind changes* by $\mathbf{M}$ on f; we often drop b from queue_b if b is clear from context.
(2) $\mathbf{Qfex}^a_{b,c} = \{\mathcal{C} \mid (\exists \mathbf{M})[\mathcal{C} \subseteq \mathbf{Qfex}^a_{b,c}(\mathbf{M})]\}$.

We usually drop the superscript in $\mathbf{Sfex}^a_{b,c}$ ($\mathbf{Qfex}^a_{b,c}$) when the superscript is 0.

3.2 Simulation Results

Proposition 15 *For all* $a, b, c \in N \cup \{*\}$, $\mathbf{Qfex}^a_{b,c} \subseteq \mathbf{Sfex}^a_{b,c} \subseteq \mathbf{Ex}^a$.

Theorem 16 $(\forall c \in N)[\mathbf{Ex}^0_{c+1} - \mathbf{Sfex}^*_{*,c} \neq \emptyset]$.

Proof. Consider the class of functions $\mathcal{C} = \{f \mid \mathrm{card}(\{x \mid f(x) \neq f(x+1)\}) \leq c+1\}$. It is easy to see that $\mathcal{C} \in \mathbf{Ex}^0_{c+1}$. It can be shown that $\mathcal{C} \notin \mathbf{Sfex}^*_{*,c}$ using a proof analogous to the proof of $\mathcal{C} \notin \mathbf{Ex}^*_c$ in Case and Smith (1983). ∎

Theorem 17 $(\forall a \in N)[\mathbf{Ex}^{a+1}_0 - \mathbf{Sfex}^a_{*,*} \neq \emptyset]$.

Proof. Follows from Proposition 15 and Theorem 3. ∎

Theorem 18 *For* $1 \leq b \leq c, c \in N$, $\mathbf{Sfex}_{*,c} \subseteq \mathbf{Qfex}_{b,2c-b}$.

The proof of the above theorem requires a very complicated simulation.[4] Before we give a proof of Theorem 18, we state the following technical Lemma 19 which will facilitate some of our proofs. The proof of Lemma 19 is straightforward.

Lemma 19 *If* $\mathcal{S} \in \mathbf{Sfex}_{b,c}$*, then there exists a learning machine* $\mathbf{M}$ *which* $\mathbf{Sfex}_{b,c}$*-identifies* $\mathcal{S}$ *and, for each* $f \in \mathcal{S}$, $\mathbf{M}$ *satisfies the four conditions given below. If* $\mathcal{S} \in \mathbf{Qfex}_{b,c}$*, then there exists a learning machine* $\mathbf{M}$ *which* $\mathbf{Qfex}_{b,c}$*-identifies* $\mathcal{S}$ *and, for each* $f \in \mathcal{S}$, $\mathbf{M}$ *satisfies the conditions, (b) to (d).*

(a) $(\forall n)[\mathbf{M}(f[n]) \neq \mathbf{M}(f[n+1]) \Rightarrow f[1+\max(\{x \leq n \mid \mathbf{M}(f[x]) \neq \mathbf{M}(f[n])\})] \subseteq \varphi_{\mathbf{M}(f[n+1])}]$;

(b) $(\forall n)[(\forall x < n)[\mathbf{M}(f[n]) \neq \mathbf{M}(f[x])] \Rightarrow f[n] \subseteq \varphi_{\mathbf{M}(f[n])} \wedge \pi_2(pad^{-1}(\mathbf{M}(f[n]))) = f[n]]$;

(c) all programs ever emitted by $\mathbf{M}$ *on* f *have domain either* N *or an initial segment of* N;

(d) $\mathbf{M}$*, fed* f*, does not emit more than* $c+1$ *distinct programs.*

Proof of Theorem 18. Suppose learning machine $\mathbf{M}$ is given. We construct a machine $\mathbf{M}'$ which behaves as follows. If $\mathbf{M}$ $\mathbf{Sfex}_{*,c}$-identifies f, then $\mathbf{M}'$ will $\mathbf{Qfex}_{b,2c-b}$-identify f. Without loss of generality, we assume that

$$\mathbf{M} \text{ satisfies all the four properties stated in Lemma 19 above} \qquad \text{(G)}$$

We say useful(τ), or sometimes τ *is useful*, iff $[\mathbf{M}(\tau) \neq ? \wedge (\forall \tau' \subset \tau)[\mathbf{M}(\tau) \neq \mathbf{M}(\tau')]]$. Let UseSeg $= \{\sigma \in \mathrm{SEG} \mid \mathrm{useful}(\sigma)\}$. Let UseProg $= \{\mathbf{M}(\sigma) \mid \mathrm{useful}(\sigma)\}$. UseSeg is clearly a recursive set, and, since $\mathbf{M}$ satisfies the second condition from Lemma 19, so is UseProg.

Let

[4] An interesting fact about the simulation is that the number of distinct programs output by the simulating machine on any input function, f, is not more than the number of distinct programs output by the machine being simulated on f.

$$\begin{aligned} W[0] &= \emptyset; \\ W[\langle p,x,l\rangle+1] &= \begin{cases} W[\langle p,x,l\rangle] \cup \{\langle p,x\rangle\} & \text{if } p \in \text{UseProg} \\ & \wedge[(\forall w < x)[\langle p,w\rangle \in W[\langle p,x,l\rangle]] \wedge \Phi_p(x) \leq l]; \\ W[\langle p,x,l\rangle] & \text{otherwise.} \end{cases} \end{aligned}$$

Remark 20 *Clearly $W[i]$'s satisfy the following properties:*

$W[0] = \emptyset$,

for all i, $W[i]$ is finite,

for all i, $W[i] \subseteq W[i+1]$,

for all i, $\text{card}(W[i+1] - W[i]) \leq 1$,

for all i,p,x, $[\langle p,x+1\rangle \in W[i] \Rightarrow \langle p,x\rangle \in W[i]]$,

for all p,x,i, $[p \notin \text{UseProg} \Rightarrow \langle p,x\rangle \notin W[i]]$,

for all p,x, $[[p \in \text{UseProg} \wedge (\forall w \leq x)[\varphi_p(w)\downarrow]] \Rightarrow (\exists i)[\langle p,x\rangle \in W[i]]]$,

for all i,p,x, $[\langle p,x\rangle \in W[i] \Rightarrow \varphi_p(x)\downarrow]$ and

the canonical index of $W[i]$ can be found effectively from i.

For each $\sigma \in \text{SEG}$ such that σ is useful we define a program $Proc(\sigma)$. $\mathbf{M}'$ will only output programs of the form $Proc(\sigma)$. We define the i/o behavior of all the programs $Proc(\sigma)$ in stages below. *For the rest of the proof only, let σ, τ and γ with or without decorations range over* UseSeg.

We first describe some of the important variables and then provide a pointer to many invariants maintained by the construction.

Variables:

S_j^σ, a finite subset of UseSeg, for each j and σ;
P_j^σ, a finite subset of UseProg, for each j and σ;
ξ_j^σ, a member of SEG, for each j and σ.

The ξ_j^σ's, once defined, never get changed. We use $\xi_j^\sigma\downarrow$ to denote the fact that ξ_j^σ gets defined and ambiguously also for the value of ξ_j^σ. The S_j^σ's and P_j^σ's may change value during the execution of stages.

Let $S_j^{\sigma,s}$ denote the value of S_j^σ at the *beginning* of stage s. Let $P_j^{\sigma,s}$ denote the value of P_j^σ at the beginning of stage s. Let

$$\text{FirstEntry}(\sigma,\tau,j) = \min(\{s \mid \sigma \in S_j^{\tau,s}\}) \tag{H}$$

Let $\varphi^s_{Proc(\sigma)}$ denote the finite initial segment of $\varphi_{Proc(\sigma)}$ defined by the beginning of stage s. Let r^σ_s denote the least element not in domain($\varphi^s_{Proc(\sigma)}$). Clearly, by (G), σ can be, uniquely and algorithmically, extracted from $\mathbf{M}(\sigma)$, so we may and do let

$$m^s_{\mathbf{M}(\sigma)} = \max(\{|\sigma|\} \cup \{1 + x \mid \langle \mathbf{M}(\sigma), x\rangle \in W[s]\}) \tag{I}$$

Some of the important invariants which are maintained by the construction are mentioned in Lemmas 21 and 23. The reader may verify that these invariants indeed hold. Intuitively, at any stage s, if the construction can maintain the invariants mentioned in Lemma 21-(a) to (g) without changing the S^τ_j's and P^τ_j's, then it does so. If the construction cannot maintain the above invariants without changing the S^τ_j's and P^τ_j's, then it changes S^τ_j's and P^τ_j's in such a way that the invariants in Lemma 21-(h) to (j) and Lemma 23 are also maintained. The reader may find it helpful to refer to at least Lemma 21 while going through the construction.

The definition of the i/o behavior of all the programs $Proc(\sigma)$ immediately follows.

```
Initialization:
    Let $i_\sigma = \text{card}(\{\mathbf{M}(\tau) \mid \tau \subseteq \sigma\})$.
    Let
        $S^\sigma_{i_\sigma} = \{\sigma\}$;
        $P^\sigma_{i_\sigma} = \{\mathbf{M}(\sigma)\}$;
        $S^\sigma_j = \emptyset$ for $j \neq i_\sigma$;
        $P^\sigma_j = \emptyset$ for $j \neq i_\sigma$;
        $\xi^\sigma_{i_\sigma} = \sigma$;
        for all $x < |\sigma|$, let $\varphi_{Proc(\sigma)}(x) = \sigma(x)$;
        for all $j \neq i_\sigma$, $\xi^\sigma_j$ is undefined at this point.
    Go to stage 0;
Begin stage $s$
    if  $W[s+1] - W[s] = \emptyset$                                   (1)
    then Go to stage $s+1$                                            (2)
    else                                                              (3)
        (* By Remark 20, card($W[s+1] - W[s]) \leq 1$. *)
        Let $\langle p, x\rangle$ be the unique elment of $W[s+1] - W[s]$;            (4)
        Let $j, \sigma$ be such that $p \in P^{\sigma,s}_j$;                           (5)
        (* By Lemma 21 such $j, \sigma$ exist and are unique *)
        (* Note that $m^s_p \geq x$. Thus by Lemma 21-(f), card($\{\mathbf{M}(\tau) \mid \tau \subseteq \varphi_p[x]\}) \leq$
          $j$. Also by Lemma 21-(d) and (g), if $x < |\xi^\sigma_j|$ then $\varphi_p(x) = \varphi_{Proc(\sigma)}(x)$.
          Moreover, by Lemma 21-(d) and (f), if $x \geq |\xi^\sigma_j|$ then card($\{\mathbf{M}(\tau) \mid \tau \subseteq$
          $\varphi_p[x]\}) = j$. *)
        if card($\{\mathbf{M}(\tau) \mid \tau \subseteq \varphi_p[x+1]\}) > j$                  (6)
```

```
then                                                                   (7)
    if $\sigma \notin S_j^{\sigma,s} \bigvee x < r_s^\sigma$                                 (8)
    then                                                               (9)
        Let $\tau$ be the element of largest length in $(S_j^{\sigma,s} - \{\sigma\})$;   (10)
        (* We selected the element of largest length so that the in-
          variants in Lemma 22 hold. This merely makes it easier to
          prove other lemmas. *)
        Let $\varphi_{Proc(\tau)}(w) = \varphi_p(w)$ for $r_s^\tau \leq w \leq x$;          (11)
        Let $S_{j+1}^{\varphi_p[x+1]} = S_{j+1}^{\varphi_p[x+1]} \cup \{\tau\}$;                    (12)
        Let $P_{j+1}^{\varphi_p[x+1]} = P_{j+1}^{\varphi_p[x+1]} \cup \{p\}$;                    (13)
        Let $S_j^\sigma = S_j^\sigma - \{\tau\}$;                                  (14)
        Let $P_j^\sigma = P_j^\sigma - \{p\}$;                                  (15)
        Go to stage $s+1$;                                           (16)
    else                                                              (17)
        Let $\varphi_{Proc(\sigma)}(w) = \varphi_p(w)$ for $r_s^\sigma \leq w \leq x$;          (18)
        Let $S_{j+1}^{\varphi_p[x+1]} = S_{j+1}^{\varphi_p[x+1]} \cup \{\sigma\}$;                    (19)
        Let $P_{j+1}^{\varphi_p[x+1]} = P_{j+1}^{\varphi_p[x+1]} \cup \{p\}$;                    (20)
        Let $S_j^\sigma = S_j^\sigma - \{\sigma\}$;                                  (21)
        Let $P_j^\sigma = P_j^\sigma - \{p\}$;                                  (22)
        Go to stage $s+1$;                                           (23)
    endif
elseif $x < r_s^\sigma \bigwedge \varphi_p(x) \neq \varphi_{Proc(\sigma)}(x)$                         (24)
then                                                                  (25)
    Let $\tau$ be the element of largest length in $(S_j^{\sigma,s} - \{\sigma\})$;       (26)
    (* We selected the element of largest length so that the invariants in
      Lemma 22 hold. This merely makes it easier to prove other lemmas.
      *)
    Let $\varphi_{Proc(\tau)}(w) = \varphi_p(w)$ for $r_s^\tau \leq w \leq x$;              (27)
    if there exist $j, \gamma$ such that $\xi_j^\gamma$ is already defined and
      $\xi_j^\gamma = \varphi_p[x+1]$                                          (28)
    (* By Lemma 21-(e) there is at most one such $j, \gamma$ *)
    (* Note that by Lemma 21-(d) $\xi_j^\tau$ has not been defined until now *)
    then                                                              (29)
    (* Note that by Lemma 21-(f), $\gamma$ cannot be the same as $\sigma$ *)
        Let $S_j^\gamma = S_j^\gamma \cup \{\tau\}$;                                  (30)
        Let $P_j^\gamma = P_j^\gamma \cup \{p\}$;                                  (31)
    else                                                              (32)
        Let $S_j^\tau = S_j^\tau \cup \{\tau\}$;                                  (33)
```

```
                Let $P_j^\tau = P_j^\tau \cup \{p\}$;                              (34)
                Let $\xi_j^\tau = \varphi_p[x+1]$;                                 (35)
            endif                                                                  (36)
            Let $S_j^\sigma = S_j^\sigma - \{\tau\}$;                              (37)
            Let $P_j^\sigma = P_j^\sigma - \{p\}$;                                 (38)
            Go to stage $s+1$;                                                     (39)
        elseif $x = r_s^\sigma$ then                                               (40)
            Let $\varphi_{Proc(\sigma)}(x) = \varphi_p(x)$;                        (41)
            Go to stage $s+1$;                                                     (42)
        endif                                                                      (43)
    endif                                                                          (44)
End stage $s$
```

Now consider the following set of lemmas.

Lemma 21 *The following invariants are maintained by the construction.*

(a) $(\forall s, j, \sigma)[\text{card}(S_j^{\sigma,s}) = \text{card}(P_j^{\sigma,s})]$.

(b) $(\forall p \in \text{UseProg})(\forall s)(\exists! \langle \tau, j\rangle)[p \in P_j^{\tau,s}]$.

(c) $(\forall \sigma)(\forall s)(\exists! \langle \tau', j'\rangle)[\sigma \in S_{j'}^{\tau',s}]$.

(d) $(\forall j, \sigma, s')$

- *(i)* $[[\xi_j^\sigma$ *remains undefined before stage* s' $] \Leftrightarrow (\forall s \le s')[S_j^{\sigma,s} = \emptyset]]$, *and*
- *(ii)* $[[\xi_j^\sigma$ *gets defined in Initialization* $] \Rightarrow S_j^{\sigma,0} = \{\sigma\}]$, *and*
- *(iii)* $[[\xi_j^\sigma$ *gets defined in stage* $s] \Rightarrow [$
  $\xi_j^\sigma = \varphi_{Proc(\sigma)}^{s+1} \wedge$
  $(\forall s' \le s)[S_j^{\sigma,s'} = \emptyset] \wedge$
  $S_j^{\sigma,s+1} = \{\sigma\} \wedge$
  $[\sigma \in S_{j'}^{\tau,s} \Rightarrow$
  $\quad [(\forall x < |\xi_j^\sigma| - 1)[\xi_j^\sigma(x) = \varphi_{Proc(\tau)}^s(x)\downarrow] \wedge$
  $\quad \xi_j^\sigma(|\xi_j^\sigma| - 1) \ne \varphi_{Proc(\tau)}^s(|\xi_j^\sigma| - 1)\downarrow \wedge$
  $\quad j' = j]]$
  $]]$, *and*
- *(iv)* $[\xi_j^\sigma$ *gets defined* $\Rightarrow \text{card}(\{\mathbf{M}(\tau) \mid \tau \subseteq \xi_j^\sigma\}) = j]$, *and*
- *(v)* $[\sigma \in \xi_j^{\tau,s} \wedge \tau \ne \sigma \Rightarrow \xi_j^\sigma$ *does not get defined before stage* $s]$, *and*
- *(vi)* $[\sigma \in \xi_j^{\tau,s} \Rightarrow (\forall j' > j)[\xi_{j'}^\sigma$ *does not get defined before stage* $s]]$.

(e) $(\forall j, j', \sigma, \sigma')[\xi_j^\sigma = \xi_{j'}^{\sigma'} \Rightarrow [j = j' \wedge \sigma = \sigma']]$.

(f) $(\forall j, s, \sigma)(\forall p \in P_j^{\sigma,s})$

(i) $\xi_j^\sigma\downarrow \wedge$

(ii) $\xi_j^\sigma \subseteq \varphi_p[m_p^s] \subseteq \varphi^s_{Proc(\sigma)} \wedge$

(iii) $\mathrm{card}(\{\mathbf{M}(\tau) \mid \tau \subseteq \varphi_p[m_p^s]\}) = j$.

(g) $(\forall s, j, \sigma)$

(i) $(\forall \tau \in S_j^{\sigma,s} - \{\sigma\})[\xi_j^\sigma = \varphi^s_{Proc(\tau)}]$.

(ii) $\sigma \in S_j^{\sigma,s} \Rightarrow [\varphi^s_{Proc(\sigma)} = \bigcup_{p \in P_j^{\sigma,s}} \varphi_p[m_p^s]]$.

(h) $(\forall s)$

(i) $[\mathrm{card}(\{\sigma \mid (\exists j, \tau)[\sigma \in S_j^{\tau,s} - S_j^{\tau,s+1}]\}) \leq 1]$.

(ii) $[\mathrm{card}(\{p \in \mathrm{UseProg} \mid (\exists j, \tau)[p \in P_j^{\tau,s} - P_j^{\tau,s+1}]\}) \leq 1]$.

(i) $(\forall p, \sigma, \tau, \tau', j, j', s)[[[\sigma \in S_j^{\tau,s} \wedge \sigma \in S_{j'}^{\tau',s+1}] \bigvee [p \in P_j^{\tau,s} \wedge p \in P_{j'}^{\tau',s+1}]] \Rightarrow j' \in \{j, j+1\}]$.

(j) $(\forall \sigma, \tau, \tau', j, s)[[\sigma \in S_j^{\tau,s} \wedge \sigma \in S_j^{\tau',s+1} \wedge \tau \neq \tau'] \Rightarrow \tau \neq \sigma]$.

PROOF.
(a), (b), (c): By induction on s. Clearly, invariants (a), (b), (c) hold for $s = 0$ (by Initialization). Also each stage s clearly maintains the invariants (the only changes to S_j^σ, P_j^σ are done via steps, (12)-(15), (19)-(22), (30)-(38), which maintain the invariants).

(h), (i), (j): By construction (see steps, (12)-(15), (19)-(22), (30)-(38)).

(d) parts (v), (vi): Follows by induction on stage number and using Lemma 21-(i) and (j). (Note that this implies that at step (35) in the construction, ξ_j^τ was not previously defined.)

(d) parts (i), (ii): Whenever ξ_j^σ is defined, the construction makes $S_j^\sigma = \{\sigma\}$.

(e) Follows using Lemma 21-(d) part (v), and the test at step (28).

(d) part (iv), (f) and (g): The proof is by induction on s, where for (d) part (iv), we consider those ξ_j^σ's which are defined before stage s.

Clearly, these invariants hold for $s = 0$. (f) part (i) follows from (a) and (d) part (i). Assume by induction that these invariants hold for $s = s'$. If we add τ to S_j^σ at stage s', then the construction makes $\varphi^{s'+1}_{Proc(\tau)} = \xi_j^\sigma$. Thus (g) part (i) holds for $s = s' + 1$. Also p is added to P_j^σ at stage s', only if, $\xi_j^\sigma = \varphi_p[m_p^{s+1}]$. This along with the fact that, if for some $p \in P_j^{\sigma,s'}$, $\varphi_p[m_p^{s'}] \subset \varphi_p[m_p^{s'+1}]$, then, either $\varphi_p[m_p^{s'+1}] \subseteq \varphi^{s'}_{Proc(\sigma)}$, or one of tests at steps (6), (24) and

(40) succeeds. In either case, it is easy to see that the invariants (d) part (iv), (g) part (ii) and (f) parts (ii) and (iii) hold for $s = s' + 1$.

(d) part (iii): If ξ_j^σ is defined at stage s then the construction makes $S_j^{\sigma,s+1} = \{\sigma\}$ and $\varphi_{Proc(\sigma)}^{s+1} = \xi_j^\sigma$. Suppose ξ_j^σ gets defined in stage s and $\sigma \in S_{j'}^{\tau,s}$ (such a τ, j' exist and are unique by part (a)). Then, $j' = j, \tau \neq \sigma$. Also by (g), $(\forall x < |\xi_j^\sigma| - 1)[\xi_j^\sigma(x) = \varphi_{Proc(\tau)}^s(x)\downarrow]$. Moreover by the test at step (24), $\xi_j^\sigma(|\xi_j^\sigma| - 1) \neq \varphi_{Proc(\tau)}^s(|\xi_j^\sigma| - 1)\downarrow$. ∎

Lemma 22 *(a) Suppose ξ_j^σ gets defined in stage s. Let $T = \{(j,\tau) \mid \xi_j^\tau\downarrow \subset \xi_j^\sigma\}$. Let $(j,\tau) \in T$ be such that $|\xi_j^\tau|$ is maximized. Then $\sigma \in S_j^{\tau,s}$.*

(b) $(\forall s)(\forall j,\sigma,\tau)[\tau \in S_j^{\sigma,s} \Rightarrow \tau \subseteq \sigma]$.

(c) $(\forall j,\sigma)(\forall \sigma')[[\xi_j^\sigma$ *and* $\xi_j^{\sigma'}$ *are both defined* $\wedge\, \xi_j^\sigma \subset \xi_j^{\sigma'}] \Rightarrow \sigma' \subseteq \sigma]$.

(d) $(\forall \sigma,\tau,\gamma,\gamma',j,j' \mid j \neq j' \vee \gamma \neq \gamma')[(\exists s,s')[\sigma \in S_j^{\gamma,s} \wedge \sigma \in S_{j'}^{\gamma',s+1} \wedge \tau \in S_j^{\gamma,s'} \wedge \tau \in S_{j'}^{\gamma',s'+1} \wedge \text{FirstEntry}(\sigma,\gamma,j) > \text{FirstEntry}(\tau,\gamma,j) \wedge \sigma \subset \tau] \Rightarrow [\text{FirstEntry}(\sigma,\gamma',j') > \text{FirstEntry}(\tau,\gamma',j')]]$.

(e) $(\forall j,\sigma,\tau)[\xi_j^\sigma\downarrow \subset \xi_j^\tau\downarrow \Rightarrow \xi_j^\tau$ *is convergently different from* $\varphi_{Proc(\sigma)}]$.

Proof.
(a) Suppose by way of contradiction that (a) does not hold. Let ξ_j^σ, be of least length, such that (a) does not hold for ξ_j^σ, and ξ_j^σ gets defined in stage s. Let τ be as in (a) for ξ_j^σ. Let j', τ' be such that $\sigma \in S_{j'}^{\tau',s}$ (such j', τ' exist and are unique by Lemma 21-(c)). By Lemma 21-(d), $j' = j$. We claim that $\tau = \tau'$, contradicting the fact that ξ_j^σ does not satisfy (a). Suppose by way of contradiction that there exists a γ such that $\xi_j^{\tau'} \subset \xi_j^\gamma\downarrow \subset \xi_j^\sigma$. Choose ξ_j^γ, minimizing $|\xi_j^\gamma|$, such that $\xi_j^{\tau'} \subset \xi_j^\gamma\downarrow \subset \xi_j^\sigma$. Then by Lemma 21-(d), for $x = |\xi_j^\gamma| - 1$, $\xi_j^\gamma(x) \neq \varphi_{Proc(\tau')}(x)\downarrow$, but, again by Lemma 21-(d), $\xi_j^\sigma(x) = \varphi_{Proc(\tau')}(x)$, a contradiction.
As a corollary to (a) we have that if $\xi_j^\sigma\downarrow \subset \xi_j^\tau\downarrow$ then ξ_j^σ gets defined before ξ_j^τ does.

(b) Using Lemma 21-(d)-(iii), Lemma 21-(g) we have that $\tau \subseteq \varphi_{Proc(\tau)}^0 \subseteq \varphi_{Proc(\tau)}^s \subseteq \varphi_{Proc(\sigma)}^s$ and $\sigma \subseteq \varphi_{Proc(\sigma)}^s$. Thus $\sigma \subseteq \tau$ or $\tau \subseteq \sigma$. (b) now follows using Lemma 25.

(c) If $\xi_j^\sigma \subset \xi_j^{\sigma'}$, then $\xi_j^{\sigma'}$ could not have been defined at Initialization (see the corollary mentioned after the proof of part (a)). (c) now follows using parts (a) and (b).

(d) Note that by Lemma 21 parts (d)-(iii), (e), and (g), $(\forall \sigma,\gamma,t,t',t'' \mid t < t' < t'')[[\sigma \in S_j^{\gamma,t} \wedge \sigma \notin S_j^{\gamma,t'}] \Rightarrow \sigma \notin S_j^{\gamma,t''}]$. (d) now follows by the choice of τ at steps (10) and (26) in the construction.

(e) Follows using part (a) and Lemma 21-(d). ∎

By Lemma 21-(a) and (h), given, τ, τ', j, j' such that $\tau \neq \tau'$ or $j \neq j'$,

$$(\exists\sigma)[\sigma \in S_j^{\tau,s} \wedge \sigma \in S_{j'}^{\tau',s+1}] \Leftrightarrow (\exists p)[p \in P_j^{\tau,s} \wedge p \in P_{j'}^{\tau',s+1}]$$

Also, τ, τ', j, j', if any, such that $[\tau \neq \tau'$ or $j \neq j']$, and $(\exists\sigma)[\sigma \in S_j^{\tau,s} \wedge \sigma \in S_{j'}^{\tau',s+1}]$ and $(\exists p)[p \in P_j^{\tau,s} \wedge p \in P_{j'}^{\tau',s+1}]$, are unique.

Thus in the statement just below of Lemma 23, for any s, if the supposition is true for $\tau, \tau', j, j', \sigma, p$, then $\tau, \tau', j, j', \sigma, p$ are unique.

Lemma 23 *Given s, suppose $\tau, \tau', j, j', \sigma, p$ are such that $[\tau \neq \tau' \bigvee j \neq j'] \wedge [\sigma \in (S_j^{\tau,s} \cap S_{j'}^{\tau',s+1}) \wedge p \in (S_j^{\tau,s} \cap S_{j'}^{\tau',s+1})]$, then the following invariants are maintained by the construction.*

(a) If $j = j'$, then

(i) $\sigma \neq \tau$, and

(ii) $\xi_j^{\tau'} = \varphi_{Proc(\sigma)}^{s+1} = \varphi_p[m_p^{s+1}]$, and

(iii) For $x = |\xi_{j'}^{\tau'}| - 1$, $[\xi_j^{\tau'}(x){\downarrow} \neq \varphi_{Proc(\tau)}^{s}(x){\downarrow}]$, and

(iv) $\tau' \subset \tau$.

(b) If $j' = j + 1$, then

(i) $\sigma \neq \tau'$, and

(ii) $\xi_{j'}^{\tau'} = \varphi_{Proc(\sigma)}^{s+1} = \varphi_p[m_p^{s+1}]$, and

(iii) For $x = |\xi_{j'}^{\tau'}| - 1$, $[\sigma \neq \tau \Rightarrow [\xi_{j'}^{\tau'}(x){\downarrow} \neq \varphi_{Proc(\tau)}^{s}(x){\downarrow}]]$.

Proof. Parts (a) (i)-(iii) and (b), are easy to prove using Lemma 21 and construction steps (6)-(41). Part (a)-(iv) follows using Lemma 22-(c). ∎

Lemma 24 *For all j, σ, τ such that $\sigma \neq \tau$*

(a) $[\xi_j^\sigma$ is defined $\wedge \tau \subseteq \xi_j^\sigma \subseteq \varphi_{\mathbf{M}(\tau)}] \Rightarrow (\exists s)[\mathbf{M}(\tau) \in P_j^{\sigma,s}]$, and

(b) $[\xi_j^\sigma$ is defined $\wedge \tau \subseteq \xi_j^\sigma \subseteq \varphi_{Proc(\tau)}] \Rightarrow (\exists s)[\tau \in S_j^{\sigma,s}]$, and

(c) $[\xi_j^\tau$ and ξ_j^σ are defined $\wedge\, \xi_j^\tau \subset \xi_j^\sigma] \Rightarrow (\exists x < |\xi_j^\sigma|)[\xi_j^\sigma(x) \neq \varphi_{Proc(\tau)}(x){\downarrow}]$.

Proof.
(a) Let $Err = \{(j, \sigma, \tau) \mid [\xi_j^\sigma$ is defined $\wedge \tau \subseteq \xi_j^\sigma \subseteq \varphi_{\mathbf{M}(\tau)}] \wedge \neg(\exists s)[\mathbf{M}(\tau) \in P_j^{\sigma,s}]\}$. Suppose by way of contradiction that Err is not empty. Choose j, σ, minimizing the length of ξ_j^σ, such that there exists a τ, $(j, \sigma, \tau) \in Err$. Choose τ such that $(j, \sigma, \tau) \in Err$ and $\max(\{|\xi_{j'}^\gamma| : (\exists s)[\mathbf{M}(\tau) \in P_{j'}^{\gamma,s}] \wedge \xi_{j'}^\gamma \subseteq \xi_j^\sigma\})$ is maximized. Let γ, j' be such that $(\exists s)[\mathbf{M}(\tau) \in P_{j'}^{\gamma,s}]$, $\xi_{j'}^\gamma \subset \xi_j^\sigma$, and $|\xi_{j'}^\gamma|$ is maximized. Let s be such that $\langle \mathbf{M}(\tau), |\xi_j^\sigma| - 1\rangle \in W[s+1] - W[s]$. If $j > j'$, then by the if

clause at step (6) of the construction in stage s, $\mathbf{M}(\tau)$ would be removed from $P_{j'}^{\gamma}$ and added to P_j^{σ}. If $j' = j$, then by the test at step (24) and by Lemma 21-(d) and 22-(a), $\mathbf{M}(\tau)$ must be removed from $P_{j'}^{\gamma}$ and added to P_j^{σ} in stage s.

(b) Proof is similar to that of (a).

(c) Follows using Lemma 21-(d) and Lemma 22-(a). ▮

Lemma 25 $(\forall \sigma, \tau, \gamma \mid \sigma \subset \tau)[(\exists s, s')[\sigma \in S_j^{\gamma,s} \wedge \tau \in S_j^{\gamma,s'}] \Rightarrow \mathrm{FirstEntry}(\tau,\gamma,j) < \mathrm{FirstEntry}(\sigma,\gamma,j)]$.

Proof. Follows by induction on the length of ξ_j^{γ}, using, Lemmas 22-(d) and 24-(b). ▮

Lemma 26

(a) For all $j > c+1$, for all σ, ξ_j^{σ} remains undefined forever.

(b) For all σ, there exists an s, j, τ such that $(\forall s' \geq s)[\sigma \in S_j^{\tau,s'}]$.

(c) For all $p \in \mathrm{UseProg}$, there exists an s, j, τ such that $(\forall s' \geq s)[p \in P_j^{\tau,s'}]$.

Proof. (a) follows from Lemma 21-(d) and the fact that $\mathbf{M}$ satisfies clause 4 in Lemma 19. (b) and (c) now follow from Lemma 23. ▮

Lemma 27 $(\forall \sigma, \tau \mid \sigma \subset \tau)[[\tau \subseteq \varphi_{Proc(\sigma)}] \Rightarrow [(\exists x)[\varphi_{Proc(\sigma)}(x)\downarrow \neq \varphi_{Proc(\tau)}(x)\downarrow] \bigvee [\varphi_{Proc(\sigma)} \subseteq \varphi_{Proc(\tau)} \wedge \varphi_{Proc(\sigma)}$ *is a finite initial segment*$]]]$.

Proof. Let σ, τ be given. Let j, γ be such that $\xi_j^{\gamma} \subseteq \varphi_{Proc(\tau)} \cap \varphi_{Proc(\sigma)}$, and $|\xi_j^{\gamma}|$ is maximized. Now, by Lemma 25, $\mathrm{FirstEntry}(\tau,\gamma,j) < \mathrm{FirstEntry}(\sigma,\gamma,j)$.

By Lemmas 24, 21-(g) and 22-(e), if $\gamma = \tau$ then, either $\varphi_{Proc(\sigma)}$ is convergently different from $\varphi_{Proc(\tau)}$ or $\varphi_{Proc(\sigma)} = \xi_j^{\tau}$.

Suppose $\gamma \neq \tau$. Clearly, since $|\sigma| < |\tau|$, $(\overset{\infty}{\forall} s)[\tau \in S_j^{\gamma}] \Rightarrow (\overset{\infty}{\forall} s)[\sigma \in S_j^{\gamma}]$. If $(\overset{\infty}{\forall} s)[\tau \in S_j^{\gamma}]$, then, clearly, $\varphi_{Proc(\sigma)} = \xi_j^{\gamma}$, which satisfies the claim in the lemma. Otherwise, let j', γ' be such that $(\overset{\infty}{\forall} s)[\tau \in S_{j'}^{\gamma',s}]$ (there exists such a j', γ' by Lemma 26). Hence, $\xi_{j'}^{\gamma'} \not\subseteq \varphi_{Proc(\sigma)}$. ▮

We now continue with the rest of the proof of Theorem 18. From Lemmas 21-(g), (h), 24, and 26, it follows that, for all f such that $\mathbf{M}$ $\mathbf{Sfex}_{*,c}$-identifies f, there exists a $\sigma \subseteq f$, such that $\varphi_{Proc(\sigma)} = f$. Moreover, by Lemma 27 $(\forall \tau \mid \sigma \subset \tau)[\varphi_{Proc(\tau)} \not\subseteq f]$.

Define $\mathbf{M}'$ as follows.

If $\mathbf{M}(f[n]) =?$, then let $\mathbf{M}'(f[n]) =?$; if $\text{card}(\{\mathbf{M}(f[i]) \mid i \le n\}) \ge c+1$, then let $\mathbf{M}'(f[n]) = \mathbf{M}(f[m])$, where $m = \min(\{j \mid \text{card}(\{\mathbf{M}(f[i]) \mid i \le j\}) \ge c+1\})$; otherwise, let $n_0 < n_1 < n_2 < \ldots < n_j \le n$ be such that the $f[n_i]$'s are useful and are the only useful segments of that form. Let $\mathbf{M}'(f[n]) = \mathit{Proc}(f[n_i])$, where $i = \max(\{k \le j \mid \text{card}(\{x < n \mid \Phi_{\mathit{Proc}(f[n_k])}(x) \le n \wedge \varphi_{\mathit{Proc}(f[n_k])}(x) \ne f(x)\}) = 0\})$. From the remarks just before the definition of $\mathbf{M}'$ it is easy to see that $\mathbf{M}'$, $\mathbf{Qfex}_{b,2c-b}$-identifies f. ▮

Theorem 28 *Let $b \le b' \le c, c \in N$. Let $c' = (\lceil \frac{c-b'}{b'-b+1} \rceil + 1) * (b-1) + c$. Then* $\mathbf{Sfex}^a_{b,c} \subseteq \mathbf{Qfex}^a_{b',c'}$.

Proof. Fix a, b, c and b'. For a given $\mathbf{M}$, let $\mathbf{F_M}$ be a machine which on input $f[n]$, behaves as follows. For a given f, $j \ge 1$, let $\text{Sat}(j)$ be true iff there exists a sequence of sets $S_0, S_1, \ldots, S_j$ such that, for all $i < j$, the following five conditions are satisfied.

(1) $\text{card}(S_i) \le b$.

(2) $S_0 = \{\mathbf{M}(f[0])\} - \{?\}$.

(3) $S_{i+1} - S_i \subseteq \{\mathbf{M}(f[i+1])\}$.

(4) $\mathbf{M}(f[i]) \in S_i \cup \{?\}$.

(5) $\text{card}(\{i < j \mid \emptyset \ne S_i \ne S_{i+1}\}) < c$.

If $\text{Sat}(n)$, then let $\mathbf{F_M}(f[n]) = \mathbf{M}(f[n])$. Otherwise, let $m = \max(\{j \mid \text{Sat}(j)\})$ and then let $\mathbf{F_M}(f[n]) = \mathbf{M}(f[m+1])$. Now consider the following cases.

Case 1: For all n, $\text{Sat}(n)$.

In this case, clearly, $\mathbf{F_M}$ $\mathbf{Qfex}^a_{b',*}$-identifies f.

Case 2: $(\exists j)[\neg\text{Sat}(j)]$.

Let n be the least j such that $\neg\text{Sat}(j)$. Note that, since $\mathbf{M}$ $\mathbf{Sfex}^a_{b,c}$-identifies f, it must be the case that $\varphi_{\mathbf{M}(f[n])} =^a f$. Thus, $\mathbf{F_M}$ $\mathbf{Qfex}^a_{b',*}$-identifies f.

From the above two cases it follows that $\mathbf{F_M}$ $\mathbf{Qfex}^a_{b',*}$-identifies f.

We now wish to prove an upper bound on the number of queue mind changes for $\mathbf{F_M}$ on functions $\mathbf{Sfex}^a_{b,c}$-identified by $\mathbf{M}$. To ease the proof of the bound, we will shortly consider some transformations on the behavior of $\mathbf{M}$ on function, f, which is $\mathbf{Sfex}^a_{b,c}$-identified by $\mathbf{M}$. These transformations depend both on the machine $\mathbf{M}$ and the function f. Let us first consider some of the invariants, maintained by the transformations. Suppose f, which is $\mathbf{Sfex}^a_{b,c}$-identified by $\mathbf{M}$ is given. Let $\mathbf{M}^f_{\text{before}}$ and $\mathbf{M}^f_{\text{after}}$ denote $\mathbf{M}$ before and after a transformation.

The following invariants would be maintained.

(a) $\mathbf{M}^f_{\text{after}}$ $\mathbf{Sfex}^a_{b,c}$-identifies f.

(b) Number of queue mind changes of $\mathbf{F}_{\mathbf{M}^f_{\text{after}}}$ on f is not less than the number of queue mind changes of $\mathbf{F}_{\mathbf{M}^f_{\text{before}}}$ on f.

Let $\mathbf{M}^f_{\text{final}}$ denote $\mathbf{M}$ after all the transformations are done.

We will later prove an upper bound on the number of mind changes of $\mathbf{F}_{\mathbf{M}^f_{\text{final}}}$ on f, for f $\mathbf{Sfex}^a_{b,c}$-identified by $\mathbf{M}$. Clearly, by the invariants mentioned above, this upper bound would also be an upper bound on the number of queue mind changes of $\mathbf{F}_{\mathbf{M}}$ on functions $\mathbf{Sfex}^a_{b,c}$-identified by $\mathbf{M}$.

We now proceed to describe our transformations. It will be easy to see that the invariants mentioned above are maintained. Let $\mathbf{M}$, and f, $\mathbf{Sfex}^a_{b,c}$-identified by $\mathbf{M}$ be given. Let Sat be defined as above.

(a) If there exists a j such that $\neg\text{Sat}(j)$, then let n be the least such j. For all $n' > n$ let $\mathbf{M}(f[n']) = \mathbf{M}(f[n])$.
(Using argument similar to that in Case 2 above, it can be seen that $\mathbf{M}$, after the above transformation still $\mathbf{Sfex}^a_{b,c}$-identifies f. Moreover the above transformation does not change the behavior of $\mathbf{F}_{\mathbf{M}}$ on f. Thus the invariants are maintained.)

(b) Let $S_0, S_1, \ldots$ be a sequence of sets witnessing that $\mathbf{M}$, $\mathbf{Sfex}^a_{b,c}$-identifies f (in the sense of Definition 12). We do not perform this transformation unless there exists a pair (p, i) such that

$$(\exists k > i)[p \in S_i - S_{i+1} \wedge p \in S_k] \tag{J}$$

Suppose (J) holds for some (p, i). Let l be such that $(\forall j)[\text{pad}(p, l) \notin S_j]$. Note that such an l must exist since there are only finitely many distinct sets in the sequence, $S_0, S_1, \ldots$.

$$\text{For each } k > i, \text{ if } \mathbf{M}(f[k]) = p, \text{ change } \mathbf{M}(f[k]) \text{ to } \text{pad}(p, l) \text{ and replace } p \text{ by } \text{pad}(p, l) \text{ in } S_k \tag{K}$$

When this is done see if there are now any *new* pair (p, i) such that, for the *new* S_j's, (J) is true. If so, pick one and repeat (K) (with l such that $(\forall j)[\text{pad}(p, l) \notin S_j]$). Continue in this fashion until no *new* (p, i) exist. This process must terminate since $\mathbf{M}$ on f has atmost c mind changes.

It is easy to see that the above transformations preserve the invariants mentioned above. Note that, due to the first transformation, for all j, $\mathbf{M}^f_{\text{final}}(f[j]) = \mathbf{F}_{\mathbf{M}^f_{\text{final}}}(f[j])$. Let $n_{\mathbf{M},f}$ denote the least value of j such that for all $j' > j$, $S_{j'} = S_j$, where S_j's are as defined in the beginning of the second transformation above.

We now prove a bound on the number of queue mind changes by $\mathbf{F}^f_{\mathbf{M}_{\text{final}}}$ on f, for f $\mathbf{Sfex}^a_{b,c}$-identified by $\mathbf{M}$.

We now define the queue sequence (in the sense of Definition 14) which will help us prove the upper bound on the number of mind changes of $\mathbf{F}_{\mathbf{M}^f_{\text{final}}}$ on f.

$$Q_i^{\mathbf{M},f} = \begin{cases} (\ ) & \text{if } i=0 \text{ and } \mathbf{F}_{\mathbf{M}^f_{\text{final}}}(f[0]) = ?; \\ (\ \mathbf{F}_{\mathbf{M}^f_{\text{final}}}(f[0])) & \text{if } i=0 \text{ and } \mathbf{F}_{\mathbf{M}^f_{\text{final}}}(f[0]) \neq ?; \\ (\mathbf{F}_{\mathbf{M}^f_{\text{final}}}(f[i])) & \text{if } i = n_{\mathbf{M},f}; \\ Q_{i-1}^{\mathbf{M},f} \diamond (\mathbf{F}_{\mathbf{M}^f_{\text{final}}}(f[i])) & \text{if } i \neq n_{\mathbf{M},f} \wedge i \neq 0 \wedge \mathbf{F}_{\mathbf{M}^f_{\text{final}}}(f[i]) \notin \text{Set}(Q_{i-1}^{\mathbf{M},f}) \wedge \\ & \text{card}(\text{Set}(Q_i^{\mathbf{M},f})) < b'; \\ (\mathbf{F}_{\mathbf{M}^f_{\text{final}}}(f[i])) & i \neq n_{\mathbf{M},f} \wedge i \neq 0 \wedge \mathbf{F}_{\mathbf{M}^f_{\text{final}}}(f[i]) \notin \text{Set}(Q_{i-1}^{\mathbf{M},f}) \wedge \\ & \text{card}(\text{Set}(Q_i^{\mathbf{M},f})) = b'; \\ Q_{i-1}^{\mathbf{M},f} & \text{otherwise.} \end{cases}$$

Let $\text{pbf}(\mathbf{M}, f) = \text{card}(\{\mathbf{M}(f[j]) \mid j < n_{\mathbf{M},f} \wedge \mathbf{M}(f[j]) \neq ?\})$. ($\text{pbf}(\mathbf{M}, f)$ is the number of distinct programs output by $\mathbf{M}$ on initial segments of f of length less than $n_{\mathbf{M},f}$).

Let $g(l) = \max(\{\text{card}(\{i \mid (\ ) \neq Q_i^{\mathbf{M},f} \neq Q_{i+1}^{\mathbf{M},f} \wedge i < n_{\mathbf{M},f}\}) \mid \text{pbf}(\mathbf{M}^f_{\text{final}}, f) \leq l\})$. It is easy to see that the number of mind changes of $\mathbf{F}_{\mathbf{M}^f_{\text{final}}}$ on f, $\mathbf{Sfex}^a_{b,c}$-identified by $\mathbf{M}$, is bounded by $g(c)+b-1$. We now calculate a bound on, $g(l)$ (for $l \leq c$). If $l \leq b'$, then clearly $g(l) = l$. Also, $g(r+b') \leq b' + g(r+b-1)$, for $r \geq 1$. Solving the recurrence we get $g(c) \leq \lceil \frac{c-b'}{b'-b+1} \rceil \cdot (b-1) + c$. Thus the number of queue mind changes of $\mathbf{M}'$ on f is bounded by $g(c) + b - 1 \leq c'$. ▮

3.3 Separation Results

In this section we prove our separation results of the form, $\mathbf{Qfex}_{b',c'} - \mathbf{Sfex}_{b,c} \neq \emptyset$, $\mathbf{Sfex}_{b',c'} - \mathbf{Sfex}_{b,c} \neq \emptyset$ and $\mathbf{Sfex}_{b',c'} - \mathbf{Qfex}_{b,c} \neq \emptyset$. To this end, we prove Claim 29 below. It's proof is long and involves several lemmas; however, once Claim 29 is available the proof of the separation results are short. Intuitively, Claim 29 extracts out the common part of the diagonalization proofs in these separation results.

But first we define a useful notion of *stacklike* sequence. Consider a finite sequence of *push, pop* and *top* operations being performed on a stack, where the i-th push operation, if any, pushes i on the stack. Assume that no two top operations occur one immediately after the other, and a push operation is *always* followed by a top operation. Further assume that, top and pop operations are performed only if the stack is non-empty. In the above, the possible sequences formed using the answers to the *top* operation are characterized by what we refer to as *stacklike* sequences just below.

A sequence $(n_1, n_2, \ldots, n_k)$ is called *stacklike* just in case the following four conditions hold:

1. $\{n_i \mid 1 \leq i \leq k\}$ forms an initial segment of N^+.
2. For $1 \leq i < j < l \leq k$, if $n_i > n_j$, then $n_i \neq n_l$.
3. For $1 \leq i < j \leq k$, if $n_i > n_j$, then there exists an l, $1 \leq l < i$ such that $n_l = n_j$.
4. For $1 \leq i < k$, $n_i \neq n_{i+1}$.

Note that any non-empty stacklike sequence starts with a 1. We say that the sequence $(n_1, n_2, \ldots, n_m)$ *matches* the stacklike sequence s iff there is a 1-1 mapping g from $\{n_i \mid 1 \leq i \leq m\}$ into N such that $(g(n_1), g(n_2), \ldots, g(n_m))$ forms a prefix of s; we say $(n_1, n_2, \ldots, n_m)$ *exactly matches* s iff $(g(n_1), g(n_2), \ldots, g(n_m))$ is the same as s.

For each stacklike sequence s, we define a class of functions, $\mathcal{C}_s$, as follows. We let proj(η) be the largest subsequence $(\pi_2 \circ \eta(i_0), \pi_2 \circ \eta(i_1), \ldots,)$ of $(\pi_2 \circ \eta(0), \pi_2 \circ \eta(1), \ldots)$ such that $(\forall j)[\pi_1 \circ \eta(i_j) = 1]$.

$\mathcal{C}_s = \{f \in \mathcal{R} \mid \text{proj}(f) \text{ matches } \ s \wedge \varphi_{\pi_2(f(\max(\{x \mid \pi_1(f(x))=1\})))} = f\}$.

For a sequence $s = (n_1, n_2, n_3, \ldots)$, let reduce($s$) = be the largest subsequence of s, such that no two consecutive elements in the sequence are the same.

Claim 29 *Given a non-empty* stacklike *sequence* $s = (n_1, n_2, \ldots, n_k)$ *and an* $\mathbf{M}$*, there exists a function* $f \in \mathcal{C}_s$*, such that, if* $\mathbf{M}$ $\mathbf{Sfex}_{b,*}$ *(respectively,* $\mathbf{Qfex}_{b,*}$*)-identifies* f*, then, the associated number of mind changes of* $\mathbf{M}$ *is lower bounded by that of a device,* $\mathbf{M}'$*, which merely outputs programs such that, reduce*$((\mathbf{M}'(f[0]), \mathbf{M}'(f[1]), \ldots))$ *exactly matches* s*.*

Proof. This proof is very complicated and requires finitary versions of the operator recursion theorem (Case (1974)). Fix b. In the rest of the proof any reference to set (queue) mind changes by a machine, refers to set$_b$ (queue$_b$) mind changes by the machine. Note that, for $f \in \mathcal{C}_s$, a device, $\mathbf{M}'$, which merely outputs programs such that reduce$((\mathbf{M}'(f[0]), \mathbf{M}'(f[1]), \ldots))$ exactly matches s, does at most $k-1$ set (queue) mind changes. Thus, if for some $f \in \mathcal{C}_s$, $\mathbf{M}$ makes at least k mind changes, then the claim is trivially satisfied. We therefore assume, without loss of generality, that $\mathbf{M}$ satisfies Lemma 19 (b) – (d) in the case that $c = k-1$.

Given a non-empty stacklike sequence s and a machine $\mathbf{M}$, by the operator recursion theorem there exists a 1-1 recursive function p such that the following holds. We will pick one of the $\varphi_{p(i)}$'s to be the diagonalizing function, f.

The diagonalization procedure works in stages. The following are some of the important variables maintained by the diagonalization procedure (the value of these variables may change during the execution of the stages).

curpos: The value of curpos, at the beginning of any stage t denotes the position in the sequence s which is being *handled* at stage t. Let curpos$_t$ denote the value of curpos at the *beginning* of stage t.

leastunused denotes a number such that, for all $i \geq$ leastunused, program $p(i)$ is available for possible use in further steps of the diagonalization.

$A[\cdot]$ (an array of programs): $A[i]$'s are certain members of the range of p. Let $\mathbf{M}'$ be a machine such that, for all g and m, $\mathbf{M}'(g[m]) = \pi_2 \circ g(\max(\{x < m \mid \pi_1 \circ g(x) = 1\}))$. Informally, the ultimate aim of the diagonalization procedure is to construct a function f such that, if $\mathbf{M}$ $\mathbf{Sfex}^a_{b,*}(\mathbf{Qfex}^a_{b,*})$-identifies f, then the set (queue) mind change complexity of $\mathbf{M}$ on f is at least as much as that of $\mathbf{M}'$. Eventually, the function computed by the final value of one of the $A[i]$'s will be such an f. Each program $A[n_i]$ extends the part of the candidate for f already constructed by $A[n_1], A[n_2], \ldots, A[n_{i-1}]$. The goal of $A[n_i]$ is to compute an extension such that the programs output by $\mathbf{M}$, on this extension, are sufficient to guarantee that $\mathbf{M}$ does no better (with respect to the set (queue) mind change complexity) than $\mathbf{M}'$. However, while $A[n_i]$ is trying to achieve its goal, it might discover that, based on the extension it has computed so far, it *may not be able* to achieve its goal. If this happens, then, for some selected j, $1 \leq j < i$, $A[n_j]$ will try to find a *new* extension, of the part of the function already constructed by $A[n_{j''}], j'' \leq j$. Also, it will then be the case that $A[n_j]$ can be successful in making *one more* of the programs output by $\mathbf{M}$ on the initial segment constructed so far by $A[n_1], \ldots, A[n_j]$ convergently *wrong*. This may *spoil* the current programs assigned to $A[n_{j''}]$, $j'' > j$. The selected j, will have the property that, $\{n_1, \ldots, n_j\} \cap \{n_{j+1}, \ldots, n_i\} = \emptyset$, and thus the programs, $A[n_1], \ldots, A[n_j]$ will not be spoiled yet. (From this point on, whenever $A[n_{j''}]$, $j'' > j$, tries to achieve *its* goal, as above, it will be assigned a *new* program). An $A[n_i]$ tries to compute its extension, as described above, in stages at the beginning of which curpos $= i$.

We say that n is *active* at stage t iff $(\exists l, l' \mid 1 \leq l < \text{curpos}_t < l' \leq k)[n_l = n_{l'} = n \wedge n < n_{\text{curpos}_t}]$.

Prog$[\cdot]$ (an array of finite sets of programs): Intuitively *Prog*$[i]$, at any time, denotes the set of programs *output* by $\mathbf{M}$, on the extension constructed by $A[i]$, which are not in *Prog*$[j]$ for some active j, and are not *already* known to be *convergently wrong.*

(∗ Given t, let $D = \{n \mid n = n_{\text{curpos}_t} \vee n \text{ is active at stage } t\}$. Then during the execution of stage t, for each pair, $n_1, n_2 \in D$, $n_1 \neq n_2$, *Prog*$[n_1]$ and *Prog*$[n_2]$ will be disjoint. ∗)

Wrong$[\cdot]$ (an array of finite sets of programs): Intuitively, at stage t, Wrong$[\text{curpos}_t]$ denotes the set of programs which have been *diagonalized* against (some diagonalization may be *undone* if, at some later stage t', $\text{curpos}_{t'}$ becomes less than curpos_t).

initseq is a boolean such that the current candidate for f is to be defined at its next argument to have π_1 of its value $= 1$ iff initseq is true.

maxcontr denotes a value $\geq$ the maximum number which has already been used for contradicting programs in Wrong$[\cdot]$.

$I_0, I_1, \ldots$: These numerical values are used to make it easier to claim the invariants maintained by the diagonalization procedure.

It may help the reader to look at the invariants in Lemma 30 while going through the diagonalization procedure.

Initially, let *Prog*$[1] = \emptyset$, $A[1] = p(0)$, leastunused $= 1$, Wrong$[1] = \emptyset$, curpos $= 1$, maxcontr $= 0$ and initseq $= TRUE$. Let $\varphi^t_{p(i)}$ denote $\varphi_{p(i)}$ defined before the *beginning* of stage

t. Let $\varphi^{t,t'}_{p(i)}$ denote $\varphi_{p(i)}$ defined by the *end* of step 3.1 in substage t' (if executed) in stage t. Go to stage 0.

```
Begin stage $t$
1. Let $y = x_t = \min(\{x \mid x \notin \text{domain}(\varphi^t_{A[n_{\text{curpos}}]})\})$.
   Let maxcontr = maxcontr + 1.
2. Go to substage 0.
3. Substage $t'$
   3.1 **if** initseq **then**
               let $\varphi_{A[n_{\text{curpos}}]}(y) = \langle 1, A[n_{\text{curpos}}]\rangle$, initseq = $FALSE$, $y = y + 1$.
       **else**
               let $\varphi_{A[n_{\text{curpos}}]}(y) = \langle 0, \text{maxcontr}\rangle$, $y = y + 1$.
       **endif**
   3.2 Let Active = $\{i \mid i < n_{\text{curpos}} \wedge (\exists j \geq \text{curpos})[n_j = i]\}$.
       **if** $(\exists i \in \textit{Active})[\mathbf{M}(\varphi_{A[n_{\text{curpos}}]}[y]) \in \textit{Prog}[i] - \text{Wrong}[\text{curpos}]]$ **then**
               Let $i$ be such that $[\mathbf{M}(\varphi_{A[n_{\text{curpos}}]}[y]) \in \textit{Prog}[i] - \text{Wrong}[\text{curpos}] \wedge i \in \text{Active}]$.
               Let $l <$ curpos be the largest number such that $n_l = i$.
               Let $d = \min(\{x \mid x \notin \text{domain}(\varphi^t_{A[i]})\})$.
               **if** $[\varphi_{\mathbf{M}(\varphi_{A[n_{\text{curpos}}]}[y])}(d)\downarrow = \varphi_{A[n_{\text{curpos}}]}(d)$ in $\leq t'$ steps ] **then**
                       (* In this case $A[n_{\text{curpos}}]$ may not be able to achieve its goal as stated
                          in the informal description before the diagonalization procedure. *)
                       (* $\varphi_{A[i]}(d)$ will be defined to be $\langle 0, \text{maxcontr}+1\rangle$ in the next stage. Thus
                          $\mathbf{M}(\varphi_{A[n_{\text{curpos}}]}[y])$ would be a *convergently wrong* program output by
                          $\mathbf{M}$ on some initial segment of $\varphi_{A[i]}[d]$. *)
                       Let Wrong[$l$] = Wrong[$l$] $\cup \{\mathbf{M}(\varphi_{A[n_{\text{curpos}}]}[y])\}$.
                       Let curpos = $l$.
                       Go to stage $t + 1$.
               **else** Go to substage $t' + 1$.
               **endif**
       **endif**
   3.3 **if** $\mathbf{M}(\varphi_{A[n_{\text{curpos}}]}[y]) \notin \text{Wrong}[\text{curpos}] \cup \{?\}$ **then**
               Let $\textit{Prog}[n_{\text{curpos}}] = \textit{Prog}[n_{\text{curpos}}] \cup \{\mathbf{M}(\varphi_{A[n_{\text{curpos}}]}[y])\}$.
               Let $z = \min(\{w \mid x_t < w \leq y)[\mathbf{M}(\varphi_{A[n_{\text{curpos}}]}[w]) = \mathbf{M}(\varphi_{A[n_{\text{curpos}}]}[y])]\})$.
               **if** $[[\varphi_{\mathbf{M}(\varphi_{A[n_{\text{curpos}}]}[y])}(z)\downarrow = \langle 0, \text{maxcontr}\rangle$ in $\leq t'$ steps $] \wedge (\forall x < z)$
                 $[\varphi_{\mathbf{M}(\varphi_{A[n_{\text{curpos}}]}[y])}(x)\downarrow = \varphi_{A[n_{\text{curpos}}]}(x)] \wedge [$ length of $s$ is greater than curpos$]]$
               (* Note that, if $\varphi_{\mathbf{M}(\varphi_{A[n_{\text{curpos}}]}[y])}(z)\downarrow$, then $(\forall x < z)[\varphi_{\mathbf{M}(\varphi_{A[n_{\text{curpos}}]}[y])}(x)\downarrow]$. *)
               **then** Go to step 4.
```

```
                (* here $A[n_{\text{curpos}}]$ is successful in achieving its goal, as mentioned in the informal
                   description before the diagonalization procedure *).
                endif
            endif
        3.4 Go to substage $t'+1$.
        End substage $t'$
    4.  if $n_{\text{curpos}+1} > n_{\text{curpos}}$ then
            let $A[n_{\text{curpos}+1}] = p(\text{leastunused})$. Let $\text{leastunused} = \text{leastunused} + 1$.
            Let $Prog[n_{\text{curpos}+1}] = \emptyset$.
        endif
    5.  if $n_{\text{curpos}} \neq n_i$, for all $i$ such that $\text{curpos} < i \leq k$ then
            Let $\text{Wrong}[\text{curpos}+1] = \text{Wrong}[\text{curpos}] \cup \{\mathbf{M}(\varphi_{A[n_{\text{curpos}}]}[y])\}$.
            Let $z$ be as in step 3.3.
            Let $\varphi_{A[n_{\text{curpos}+1}]}(x) = \varphi_{A[n_{\text{curpos}}]}(x)$ for each $x < z$ such that $\varphi_{A[n_{\text{curpos}+1}]}(x)$ has not
              been defined till now.
            Let $I_{\text{curpos}} = z$.
        else
            For each $x < y$, such that $\varphi_{A[n_{\text{curpos}+1}]}(x)$ has not been defined till now,
            let $\varphi_{A[n_{\text{curpos}+1}]}(x) = \varphi_{A[n_{\text{curpos}}]}(x)$.
            Let $\text{Wrong}[\text{curpos}+1] = \text{Wrong}[\text{curpos}]$.
            Let $I_{\text{curpos}} = y$.
        endif
    6.  Let $\text{curpos} = \text{curpos} + 1$.
        Let $\text{initseq} = TRUE$.
        Go to stage $t+1$.
End stage $t$
```

Let $Prog^t[n]$ denote the value of $Prog[n]$ at the *beginning* of stage t. For i such that $1 \leq i \leq k$, let last_i^t denote the highest numbered stage, $t' \leq t$, if any, such that $\text{curpos}_{t'} = i$. Let $I_0 = -1$. Let $\text{Wrong}^t[i]$ denote the value of $\text{Wrong}[i]$ at the *beginning* of stage t.

Lemma 30 *For all t, t' such that, substage t' of stage t is executed, the following holds at the end of step 3.1 in substage t' of stage t.*

(a) For each $i < \text{curpos}_t - 1$, $I_i < I_{i+1}$.

(b) For each i such that $1 \leq i < \text{curpos}_t$, $\text{Wrong}[i] \subseteq \text{Wrong}[i+1]$.

(c) For each i such that $1 \leq i < \text{curpos}_t - 1$, $\varphi_{A[n_i]}[I_i] \subseteq \varphi_{A[n_{i+1}]}[I_{i+1}]$. Also, if $\text{curpos}_t > 1$, then $\varphi_{A[n_{\text{curpos}_t - 1}]}[I_{\text{curpos}_t - 1}] \subseteq \varphi_{A[n_{\text{curpos}_t}]}$.

(d) *For n such that n is active at stage t, let $l_n = \max(\{i \mid i \leq \text{curpos}_t \wedge n_i = n\})$. Then* $\text{domain}(\varphi^t_{A[n]}) = \{x \mid x < I_{l_n}\}$.

(e) *For each i such that $1 \leq i < \text{curpos}_t$, there exists an $x \in \{I_{i-1}+1, \ldots, I_i\}$ such that* $\mathbf{M}(\varphi_{A[\text{curpos}_t]}[x]) \in \mathit{Prog}^{1+\mathit{last}^t_i}[n_i] - \text{Wrong}^t[i]$.

(f) *For each i such that $1 \leq i < \text{curpos}_t$ and $n_i \neq n_{\text{curpos}_t}$, if n_i is not active at stage t, then $\mathbf{M}(\varphi_{A[n_{\text{curpos}_t}]}[I_i]) \in (\mathit{Prog}^{1+\mathit{last}^t_i}[n_i] - \text{Wrong}^t[i])$ and $\varphi_{\mathbf{M}(\varphi_{A[n_{\text{curpos}_t}]}[I_i])}(I_i)\downarrow \neq \varphi_{A[n_{\text{curpos}_t}]}(I_i)\downarrow$ and $\mathbf{M}(\varphi_{A[n_{\text{curpos}_t}]}[I_i]) \in \text{Wrong}^t[i+1]$.*

(g) *For each i and i' such that $[1 \leq i \leq i' \leq \text{curpos}_t \wedge n_i < n_{i'}]$, if there exists an $i'' \geq i'$ such that $n_i = n_{i''}$, then $\mathit{Prog}^{\mathit{last}^t_{i'}+1}[n_i] \cap \mathit{Prog}[n_{i'}] = \emptyset$.*

(h) *For each $n < n_{\text{curpos}_t}$, such that n is active at stage t, $\varphi^t_{A[n]} \subseteq \varphi^t_{A[n_{\text{curpos}_t}]}$.*

(i) *For n active at stage t, let $l_n = \max(\{i \mid i \leq \text{curpos}_t \wedge n_i = n\})$. For n active at stage t, proj$(\varphi^t_{A[l_n]})$ exactly matches $(n_1, \ldots, n_{l_n})$. Moreover, proj$(\varphi^{t,t'}_{A[\text{curpos}_t]})$ exactly matches $(n_1, \ldots, n_{\text{curpos}_t})$.*

(j) *For each i such that $1 \leq i < \text{curpos}_t - 1$,* $\{\mathbf{M}(\varphi_{A[n_{\text{curpos}_t}]}[x]) \mid I_i < x \leq I_{i+1}\} \subseteq \{\mathit{Prog}^{1+\mathit{last}^t_{i+1}}[n_j] \mid 1 \leq j \leq i+1 \wedge (\exists j' \mid i+1 \leq j' \leq k)[n_{j'} = n_j]\} \cup \text{Wrong}^t[i+1]$.

Proof.

We will prove the invariants by induction on the stages/substages which are executed. Clearly, the invariants hold at the end of step 3.1 of substage 0 in stage 0. Suppose the invariants in the lemma hold after step 3.1 in substage t'_1 of stage t_1. Suppose that the next substage executed after substage t'_1 of stage t_1 is substage t'_2 of stage t_2. We show that the invariants of the lemma hold after step 3.1 of substage t'_2 of stage t_2.

Case 1: In substage t'_1 of stage t_1, the first and second **if** statements of step 3.2 succeed.

In this case clearly, $t_2 = t_1 + 1$, $t'_2 = 0$ and $\text{curpos}_{t_2} < \text{curpos}_{t_1}$. Let l be as in step 3.2 of substage t'_1 of stage t_1. Also, since n_l was active at stage t_1 and for all j such that $l < j \leq \text{curpos}_{t_1}$, $n_j \neq n_l$, we have $\{n_j \mid 1 \leq j \leq l\} \cap \{n_j \mid l < j \leq \text{curpos}_{t_1}\} = \emptyset$. Thus the elements in $\{n_j \mid 1 \leq j \leq l\} - \{n_l\}$ which are active at stage t_1 would exactly be the same as the n_j's which are active at stage t_2.

From the above observations, it is easy to see that (a)-(j) hold after step 3.1 in substage t'_2 of stage t_2.

Case 2: In substage t'_1 of stage t_1, the first **if** statement of step 3.2 fails and the first and second **if** statements of step 3.3 succeed.

In this case $t_2 = t_1 + 1$ and $\text{curpos}_{t_2} = \text{curpos}_{t_1} + 1$. After step 3.1 in substage t'_2 of stage t_2:

(a), (b), (c), (d) and (h) hold because of the definition of $I_{\mathrm{curpos}_{t_1}}$, Wrong[curpos$_{t_2}$] and $\varphi_{A[n_{\mathrm{curpos}_{t_2}}]}$ in step 5 of substage t_1' in stage t_1 and the fact that the invariants were satisfied after step 3.1 of substage t_1' of stage t_1.

(f) holds because of the first assignment statement in step 3.3 of substage t_1' in stage t_1, the value assigned to $I_{\mathrm{curpos}_{t_1}}$ and Wrong[curpos$_{t_2}$], in step 5 (of stage t_1) and the fact that invariants were satisfied after step 3.1 of substage t_1' of stage t_1.

(e) and (j) hold because of the first assignment statement in step 3.3 of substage t_1' in stage t_1, and the fact that the invariants were satisfied after step 3.1 of substage t_1' of stage t_1.

(g) holds because of step 4, the check at the first **if** statement in step 3.1 of substage t_1' in stage t_1, and the fact that the invariants were satisfied after step 3.1 of substage t_1' of stage t_1.

(i) holds because of the fact that initseq is made true in step 6 of stage t_1, and the fact that the invariants were satisfied after step 3.1 of substage t_1' of stage t_1.

Case 3: Not case 1 or 2.

It is easy to see that (a)-(j) will hold after step 3.1 in substage t_2' of stage t_2. ∎

Lemma 31 *For all $i \leq k$, the number of stages at the beginning of which* curpos $= i$ *is finite.*

Proof. By induction on i. The number of stages for which curpos $= 1$ is at most $k+1$, since each time a new stage with curpos $= 1$ is started (except possibly the first time) the number of programs in Wrong[1] is increased by 1. Since the number of programs output by $\mathbf{M}$ on any initial sequence is bounded by k, the cardinality of Wrong[i] is bounded by k. Thus the number of stages with curpos $= 1$ is bounded by $k+1$. Similarly it can be shown that, if after stage t curpos is always $> i$, then the number of stages $> t$ for which curpos $= i+1$ is bounded by $k+1$. The lemma follows. ∎

As a corollary to above lemma we have that

Lemma 32 *The number of stages which are executed is finite.*

Let the last stage which is executed be t.

Let $f = \varphi_{A[n_{\mathrm{curpos}_t}]}$. Clearly f is total and in $\mathcal{C}_s$ (using Lemma 30-(i)). Suppose $\mathbf{M}$ $\mathbf{Sfex}_{b,*}$($\mathbf{Qfex}_{b,*}$)-identifies f. We give an argument for lower bounding the number of set (queue) mind changes by $\mathbf{M}$ on f,

Now, if curpos$_t \neq k$, then $\mathbf{M}$ does not $\mathbf{Fex}_*$ identify f (otherwise one of step 3.2 or 3.3 would eventually force a change of stage). Thus curpos$_t = k$.

For $i \in \{1, \cdots, k-1\}$, *let*

t_i denote the last stage at the beginning of which curpos $= i$,

P_i be the value of $Prog[n_i]$ at the end of stage t_i, and

W_i denote the value of Wrong$[i]$ at the end of stage t_i.

Let $t_k = t$, $P_k =$ the limiting value of $Prog[n_k]$ and $W_k =$ the limiting value of Wrong$[k]$. Let $I_1, I_2, \ldots, I_{k-1}$ be as in stage t.

Note that P_k is non-empty and for all but finitely many j, $\mathbf{M}(f[j]) \in P_k - W_k$ (otherwise, either, one of step 3.2 or 3.3 would eventually force a stage change, or, $\mathbf{M}$ does not $\mathbf{Sfex}_{b,*}$ ($\mathbf{Qfex}_{b,*}$)-identify f). Let $I_0 = -1$ and $I_k = \infty$.

We will now argue that the number of set (queue) mind changes for $\mathbf{M}$ on f is lower bounded by that of the machine $\mathbf{M}'$, which just outputs a sequence of programs such that reduce$(\mathbf{M}'(f[0]), \mathbf{M}'(f[1]), \ldots) = s$. For this purpose it is convenient to consider set (queue) mind change complexity of infinite sequences. Let α range over infinite sequences over $N \cup \{?\}$. We let $\alpha(i)$ denote the i-th element of α. For each α, let $\mathbf{M}_\alpha$ be such that $\mathbf{M}_\alpha(f[j]) = \alpha(j)$ (here $\mathbf{M}_\alpha$ may not be an algorithmic device; however, for the definition below this is not significant). We define the *set (queue) mind change complexity* of $\alpha =$ the set (queue) mind change complexity of $\mathbf{M}_\alpha$ on f. When we say "let $S_0, S_1, \ldots (Q_0, Q_1, \ldots)$ be the sets (queues) witnessing that set (queue) mind change complexity of α is X ($\leq X$)" we mean "let $S_0, S_1, \ldots (Q_0, Q_1, \ldots)$ be the sets (queues) witnessing that the set (queue) mind change complexity of $\mathbf{M}_\alpha$ on f is X ($\leq X$), in the sense of Definition 12 (Definition 14)."

Let $\alpha = (\mathbf{M}(f[0]), \mathbf{M}(f[1]), \ldots)$. Below we will successively change α, each change not increasing the set (queue) mind change complexity of α, such that, for the final α obtained, reduce$(\alpha) = s$. This implies that the set (queue) mind change complexity of $\mathbf{M}$ on f is lower bounded by that of $\mathbf{M}'$, and thus proves the claim.

Note that by Lemma 30 the following holds.

Lemma 33

(a) For each $i < k$, $I_i < I_{i+1}$.

(b) For each i, such that $1 \leq i < k$, $W_i \subseteq W_{i+1}$.

(c) For each i, $1 \leq i \leq k$, there exists an $x \in \{I_{i-1}+1, \ldots, I_i\}$, such that $\alpha(x) \in P_i - W_i$.

(d) For each i such that $1 \leq i < k$, if, for all $j > i$, $n_j \neq n_i$, then $\alpha(I_i) \in P_i \cap W_{i+1}$.

(e) For all j, j' such that $[1 \leq j \leq j' \leq k \wedge n_j < n_{j'}]$, if there exists a $j'' \geq j'$ such that $n_j = n_{j''}$, then $P_j \cap P_{j'} = \emptyset$.

(f) For $i < k-1$, $\{\alpha(I_i+1), \ldots, \alpha(I_i)\} \subseteq \bigcup_{\{j \mid 1 \leq j \leq i+1 \wedge (\exists j' \geq i+1)[n_j = n_{j'}]\}} P_j \cup W_{i+1}$.

(g) $(\overset{\infty}{\forall} x)[\alpha(x) \notin W_k]$.

(h) $(\forall i, j \mid 1 \leq i < j \leq k)[n_i = n_j \Rightarrow P_i \subseteq P_j]$.

We will now consider four successive transformations of α. The reader will see that each of these transformations does not increase the number of set (queue) mind changes of α.

Begin transformation T1

Let $x > I_{k-1}$ be such that $\alpha(x) \in P_k - W_k$ (there exists such an x by Lemma 33-(c)). For each $y > I_{k-1}$, let $\alpha(y) = \alpha(x)$.

(* Since, for all but finitely many z, $\alpha(z) \neq \alpha(I_{k-1})$, this does not increase the number of set (queue) mind changes of α. *)

End transformation T1

It is easy to see that,

Lemma 34 *After transformation T1, α, in addition to satisfying Lemma 33, also satisfies: for each $x > I_{k-1}$, $\alpha(x) = \alpha(I_{k-1} + 1) \in P_k$.*

Begin transformation T2

For each pair i, x, such that

$$1 \leq i \leq k - 1 \wedge I_i < x \leq I_{i+1} \wedge \alpha(x) \in W_i, \qquad \text{(L)}$$

let $y_{i,x}$ be such that $I_i < y_{i,x} \leq I_{i+1} \wedge \alpha(y_{i,x}) \notin W_i$ and $ABS(y_{i,x} - x)$ is minimized, and, then, let $\alpha(x) = \alpha(y_{i,x})$ (note that by Lemma 33-(c) there exists a y such that $I_i < y \leq I_{i+1} \wedge \alpha(y) \notin W_i$).

(* Note that this does not increase the set (queue) mind change complexity of α. *)

End transformation T2

It is easy to see that,

Lemma 35 *After transformation T2, α, in addition to satisfying Lemma 33, also satisfies*
(i) for each $x > I_{k-1}$, $\alpha(x) = \alpha(I_{k-1} + 1) \in P_k$, and
(ii) for all i, x such that $1 \leq i \leq k - 1$ and $x > I_i$, $\alpha(x) \notin W_i$.

For $n \in \{n_1, \ldots, n_k\}$, let $\text{final}_n = \max(\{i \mid 1 \leq i \leq k \wedge n_i = n\})$. For $n \in \max(\{n_1, \ldots, n_k\} - \{n_k\})$, let $E_n = \alpha(I_{\text{final}_n})$; note that $E_n \in W_{\text{final}_n+1}$. Let $E_{n_k} = \alpha(I_{k-1} + 1)$.

Note that, by aid of Lemma 33-(d), $E_{n_r} = E_{n_{r'}}$ iff $n_r = n_{r'}$. Also note that by Lemma 35, for each j, $1 \leq j < k$, $E_{n_j} = \alpha(I_{\text{final}_{n_j}})$, and for all $x > I_{\text{final}_{n_j}}$, $\alpha(x) \neq E_{n_j}$.

For $i \in \{1, \ldots, k - 1\}$, the following seven invariants will be true, in transformation T3 below, after the execution of the iteration of the **for** loop with $j = i$. For $i = 0$ the invariants are true before the start of the transformation T3.

The invariants:

(i) For each i' such that $1 \leq i' \leq i$, $E_{n_{i'}} \in \{\alpha(x) \mid I_{i'-1} < x \leq I_{i'}\}$.

(ii) $(\forall i' \mid 1 \leq i' \leq k)[E_{n_{i'}} \in \{\alpha(x) \mid I_{\text{final}_{n_{i'}}-1} < x \leq I_{\text{final}_{n_{i'}}}\}]$.

(iii) $(\forall j, j', j'' \mid 1 \leq j < j' < j'' \leq k)[[n_j = n_{j''} \neq n_{j'}] \Rightarrow [E_{n'_j} \notin \{\alpha(x) \mid I_{j-1} < x \leq I_j\}]]$.

(iv) For each i' such that $i < i' \leq k$, there exists an $x \in \{I_{i'-1}+1, \ldots, I_{i'}\}$ such that $\alpha(x) \in P_i - W_i$.

(v) Lemma 33 (a), (b), (d), (e), (g) and (h) are satisfied.

(vi) For each $x > I_{k-1}$, $\alpha(x) = \alpha(I_{k-1}+1) \in P_k$.

(vii) For all i, x such that $1 \leq i \leq k-1$ and $x > I_i$, $\alpha(x) \notin W_i$.

In transformation T3 and T4

```
“Insert $q$ after $x$ in $\alpha$” means perform the following operations in sequence:
    for each $y \leq x$, let $\alpha'(y) = \alpha(x)$;
    let $\alpha'(x+1) = q$;
    for each $y > x$, let $\alpha'(y+1) = \alpha(y)$;
    let $\alpha = \alpha'$;
    for each $j, 1 \leq j \leq k$, if $I_j > x$, then let $I_j = I_j + 1$.
```

Begin transformation T3

```
for $j = 1$ to $k-1$ do
if $E_{n_j} \notin \{\alpha(x) \mid I_{j-1} < x \leq I_j\}$ then perform the transformation given below (which
  ensures that after the transformation $E_{n_j} \in \{\alpha(x) \mid I_{j-1} < x \leq I_j\}$).
    Let $S_i$'s ($Q_i$'s) be such that the number of set (queue) mind changes of $\alpha$ is minimized.
      Let $R_i = S_i$ (respectively, $Set(Q_i)$) below.
    Case 1: For some $x$, $I_{j-1} < x \leq I_j$, $E_{n_j} \in R_x$.
        Let $x$ be such that $I_{j-1} < x \leq I_j$ and $E_{n_j} \in R_x$. Insert $E_{n_j}$ after $x-1$ in $\alpha$.
        (* Note that this does not increase the number of set (queue) mind changes for
          $\alpha$. *)
    Case 2: Not case 1.
        Let $w \in P_{n_j}$ be such that $w \in \{\alpha(x) \mid I_{j-1} < x \leq I_j\}$. Let $x_w$ be such that
          $I_{j-1} < x \leq I_j$ and $w = \alpha(x_w)$.
        Case 2.1: There does not exist a $y > I_{\text{final}_{n_j}}$ such that $w = \alpha(y)$ or there exists
          an $x$ such that $x_w < x \leq I_{\text{final}_{n_j}}$ and either $w \notin R_x$ or we are dealing with
          queue mind changes, $Q_x = Q_{x-1} \diamond \alpha(x)$, and $\alpha(x) = w$.
                Select such an $x$. For all $y < x$, such that $\alpha(y) = w$, let $\alpha(y) = E_{n_j}$.
```

```
            (* Note that this does not increase the number of set (queue) mind
               changes of $\alpha$. *)
        Case 2.2: Not case 2.1.
            Note that in this case, there exist $x$ and $y$ such that $x \leq I_j < y \leq I_{\text{final}_j}$
               and
               $(\forall z \mid x \leq z \leq y)[w \in R_z] \wedge (\forall z \mid x \leq z < y)[E_{n_j} \notin R_z] \wedge \alpha(y) = E_{n_j}$.   (M)
               Let $x, y$ be such that they satisfy (M). For all $z < y$ such that
               $\alpha(z) = w$, let $\alpha(z) = E_{n_j}$, and, then, insert $w$ after $y$ in $\alpha$.
            (* Note that this does not increase the number of set mind changes of
               $\alpha$. For queue mind change complexity note that the above transfor-
               mation interchanges the position of $w$ and $E_{n_j}$ in $Q_y$, which does not
               increase the number of queue mind changes, since, for all $z > I_{\text{final}_j}$,
               $E_{n_y} \neq \alpha(z)$. *)
    endif
    endfor
End transformation T3.
```

It is easy to see that the seven invariants stated just before the transformation hold. The following lemma follows from these invariants holding.

Lemma 36 *After transformation T3, α satisfies each of the following.*

(a) For each j, $1 \leq j \leq k$, $E_{n_j} \in \{\alpha(x) \mid I_{j-1} < x \leq I_j\}$.

(b) $(\forall j, j', j'' \mid 1 \leq j < j' < j'' \leq k)[[n_j = n_{j''} \neq n_{j'}] \Rightarrow [E_{n'_j} \notin \{\alpha(x) \mid I_{j-1} < x \leq I_j\}]]$.

(c) $(\forall j \mid 1 \leq j \leq k)[E_{n_j} \notin \{\alpha(x) \mid I_{\mathit{final}_{n_j}} < x\}]$.

Note that the above lemmas establish Claim 29 for set mind change. This is so because there exists a subsequence α' of α such that reduce(α) $= s$. For the queue mind change case we need some more transformations.

We assume, from now on, that $\{\alpha(x) \mid x \in N\}$ consists only of elements from $\{E_{n_1}, E_{n_2}, \ldots, E_{n_k}, ?\}$ (the other elements in α can be deleted, without increasing the queue mind change complexity of α). In transformation T4 below

"delete x-th element of α" means perform the following operations in sequence:

```
    for each $y < x$, let $\alpha'(y) = \alpha(x)$;
    for each $y \geq x$, let $\alpha'(y) = \alpha(y + 1)$;
    let $\alpha = \alpha'$;
    for each $j, 1 \leq j \leq k$, if $I_j \geq x$, then let $I_j = I_j - 1$.
```

Let $Prop(i)$ be the property:

$$(\exists x \mid I_{i-1} < x \leq I_i)[\alpha(x) \neq E_{n_i}]. \tag{N}$$

```
Begin transformation T4
    **while** there exists an $i$ such that $Prop(i)$ **do**
        (* Note that $\{E_{\mathrm{final}_{n'_j}} \mid j' \leq k \wedge \mathrm{final}_{n'_j} \leq \mathrm{final}_{n_i}\} \cap \{\alpha(x) \mid x > I_{\mathrm{final}_{n_i}}\} = \emptyset$. *)
        Let $i$ be such that $Prop(i)$ holds with $\mathrm{final}_{n_i}$ minimized.
        Let $Q_0, Q_1, \ldots$ witness that the queue mind change complexity of $\alpha$ is $X$, where $X$ is
          the queue mind change complexity of $\alpha$.
        Let $Ql = Q_{I_{\mathrm{final}_{n_i}}}$.
        For all $x$ such that there exists a $i', n_{i'} = n_i$, $I_{i'-1} < x \leq I_{i'}$ and $\alpha(x) \neq E_{n_i}$ delete the
          $x$-th element of $\alpha$ (in decreasing order of such $x$'s).
        Let $S = \{x \mid x$-th element is deleted in the previous step and $Q_{x-1} \neq Q_x\}$. Let
          $i_1 < i_2 < \ldots < i_{\mathrm{card}(S)}$ be such that $S = \{i_1, i_2, \ldots, i_{\mathrm{card}(S)}\}$. For each $i \in S$, let
          $e_i =$ the last element in the queue $Q_i$. Let $Q' = (e_{i_1}, e_{i_2}, \ldots, e_{i_{\mathrm{card}(S)}})$.
        (* $Q'$ is formed from the deleted elements of $\alpha$ which caused a "mind change" in the
          sequence of queues $Q_0, Q_1, \ldots$ *)
        **if** $n_i \neq n_k$ **then**,
              Insert, after $I_{\mathrm{final}_{n_i}}$ in $\alpha$, all the elements of $Q'$ starting from the rear end and
                proceeding to the front.
        **endif**
        (* Using the note mentioned at the beginning of the body of this while loop, it is easy
          to see that the body of the while loop does not increase the number of mind changes.
          *)
    **endwhile**
End transformation T4
```

Lemma 37 *After transformation T4, α satisfies:*
for each i, x such that $1 \leq i \leq k$ and $I_{i-1} < x \leq I_i$, $[I_{i-1} < I_i \wedge \alpha(x) = E_{n_i}]$.

Note that the above lemma implies that after the transformations reduce$(\alpha) = s$. Thus the requirements for satisfying Claim 29 are achieved. ∎

Theorem 38 *Suppose $2 \leq i \leq c$, $1 \leq j \leq c$ and $c \in N$.*
$\mathbf{Sfex}_{i,c} - \mathbf{Qfex}_{j,(\lfloor\frac{c-1}{j+i-2}\rfloor*(i-1)+c-1+([c-\lfloor\frac{c-1}{j+i-2}\rfloor*(j+i-2)]\dot{-}j))} \neq \emptyset.$

Proof. We prove the theorem by using Claim 29. Let $Seq(l, i, j)$ be the sequence $(1, l, l+1, \ldots, l+i-3, l+i-2, \ldots, l+j+i-3, l+i-3, l+i-4, \ldots, l)$. This sequence consists of 1, followed by a sequence of $i-2$ numbers, followed by a sequence of j numbers, followed by the reverse of the first sequence of $i-2$ numbers. Let $r = \lfloor \frac{c-1}{j+i-2} \rfloor$. Let $m = ([c - \lfloor \frac{c-1}{j+i-2} \rfloor * (j+i-2)] \dot{-} j)$. If $m > 0$, then let $s' = Seq(\lfloor \frac{c-1}{j+i-2} \rfloor * (j+i-2)+2, m+1, j) \diamond (1, c+1)$; else, let $s' = (1, \lfloor \frac{c-1}{j+i-2} \rfloor * (j+i-2)+2, \ldots, c+1)$. Let $s = Seq(2, i, j) \diamond Seq(2+1*(j+i-2), i, j) \diamond \ldots \diamond Seq(2+(r-1)*(j+i-2), i, j) \diamond s'$. The theorem follows by observing that $\mathcal{C}_s \in \mathbf{Sfex}_{i,c}$ and the fact that any machine which $\mathbf{Qfex}_{j,*}$-identifies $\mathcal{C}_s$ can do no better, with respect to queue mind changes, on some $f \in \mathcal{C}_s$, than outputting a sequence exactly matching s. ▮

Theorem 39 *Suppose* $1 \le j \le i \le c$, $c \in N$. $\mathbf{Sfex}_{i,c} - \mathbf{Sfex}_{j,(\lfloor \frac{c-1}{i-1} \rfloor *(i-j)+c-1+([c-\lfloor \frac{c-1}{i-1} \rfloor *(i-1)] \dot{-} j))} \neq \emptyset$.

Proof. We prove the theorem by using Claim 29. Let $Seq(l, i)$ be the sequence $(1, l, l+1, \ldots, l+i-3, l+i-2, l+i-3 \ldots, l)$ This sequence consists of 1, followed by a sequence of $i-2$ numbers, followed by a number, followed by the reverse of the sequence of $i-2$ numbers. Let $r = \lfloor \frac{c-1}{i-1} \rfloor$. Let $m = ([c - \lfloor \frac{c-1}{i-1} \rfloor * (i-1)] \dot{-} j)$. If $m > 0$, then let $s' = Seq(\lfloor \frac{c-1}{i-1} \rfloor * (i-1)+2, m+j) \diamond (1, c+1)$; else, let $s' = (1, \lfloor \frac{c-1}{i-1} \rfloor * (i-1)+2, \ldots, c+1)$. Let $s = Seq(2, i) \diamond Seq(2+1*(i-1), i) \diamond \ldots \diamond Seq(2+(r-1)*(i-1), i) \diamond s'$. The theorem follows by observing that $\mathcal{C}_s \in \mathbf{Sfex}_{i,c}$ and the fact that any machine which $\mathbf{Sfex}_{j,*}$-identifies $\mathcal{C}_s$ can do no better, with respect to set mind changes, on some $f \in \mathcal{C}_s$, than outputting a sequence exactly matching s. ▮

A complicated modification of the diagonalization in the proof of Claim 29 also gives the following:

Theorem 40 *Let* $1 \le j < i \le c$, $c \in N$. $\mathbf{Qfex}_{i,c} - \mathbf{Sfex}_{j,(\lfloor \frac{c-1}{i-1} \rfloor *(i-j)+c-1+([c-\lfloor \frac{c-1}{i-1} \rfloor *(i-1)] \dot{-} j))} \neq \emptyset$.

As corollaries to the above theorems we have:

Corollary 41 *For all* $c \in N, c \ge 1$, $\mathbf{Qfex}_{1,c} \subset \mathbf{Qfex}_{2,c} \subset \ldots \subset \mathbf{Qfex}_{c,c} = \mathbf{Qfex}_{c+1,c} = \ldots$.

Corollary 42 *For all* $c \in N, c \ge 1$, $\mathbf{Sfex}_{1,c} \subset \mathbf{Sfex}_{2,c} \subset \ldots \subset \mathbf{Sfex}_{c,c} = \mathbf{Sfex}_{c+1,c} = \ldots$.

Corollary 43 *For all* $b, c \in N$, $2 \le b < c$, $\mathbf{Qfex}_{b,c} \subset \mathbf{Sfex}_{b,c}$.

Corollary 44 *For* $c \in N^+$, $\mathbf{Qfex}_{2,c} - \mathbf{Ex}_{2c-2} \neq \emptyset$.

4 Program Size Restrictions on Vacillatory Function Identification

Definition 45 Suppose $a \in N \cup \{*\}$ and $b \in N^+ \cup \{*\}$.
(i) A learning machine, $\mathbf{M}$, $\mathbf{Mfex}_b^a$-*identifies* a class of functions $\mathcal{S}$ just in case $(\exists h \in \mathcal{R})(\forall f \in \mathcal{S})(\exists D \mid \mathrm{card}(D) \leq b)[\mathbf{M}(f)\Downarrow = D \wedge (\forall i \in D)[\varphi_i =^a f \wedge i \leq h(\mathrm{MinProg}_\varphi(f))]]$.
(ii) $\mathbf{Mfex}_b^a = \{\mathcal{S} \mid (\exists \mathbf{M})[\mathbf{M}\ \mathbf{Mfex}_b^a\text{-identifies } \mathcal{S}]\}$.

h in Definition 45 plays the role of a *computable* amount by which the final programs can be larger than minimal size. This size restriction of course does not hold in general, but it is not as severe as requiring that the final programs be strictly minimal size. Mathematically, $\mathbf{Mfex}_b^a$-identification is well-behaved. For example, it is *independent* of the choice of acceptable programming system used to interpret a machine's conjectures. We also denote $\mathbf{Mfex}_1^a$ by $\mathbf{Mex}^a$.

Definition 46 Suppose $a \in N \cup \{*\}$.
(i) A learning machine, $\mathbf{M}$, $\mathbf{Mbc}^a$-*identifies* a class of functions $\mathcal{S}$ just in case $(\exists h \in \mathcal{R})(\forall f \in \mathcal{S})(\overset{\infty}{\forall} n)[\varphi_{\mathbf{M}(f[n])} =^a f \wedge \mathbf{M}(f[n]) \leq h(\mathrm{MinProg}(f))]$.
(ii) $\mathbf{Mbc}^a = \{\mathcal{S} \mid (\exists \mathbf{M})[\mathbf{M}\ \mathbf{Mbc}^a\text{-identifies } \mathcal{S}]\}$.

Proposition 47 $(\forall a \in N)[\mathbf{Mfex}_*^a = \mathbf{Mbc}^a]$.

Proof. Follows from the fact that for each h, $f \in \mathcal{R}$ there are only finitely many programs $\leq h(\mathrm{MinProg}(f))$. ▮

As promised, the next theorem answers an open question in Chen (1981).

Theorem 48 $(\forall a \in N)[\mathbf{Mfex}_*^a = \mathbf{Mex}^a]$.

Proof of Theorem 48.

Suppose $\mathbf{M}$ $\mathbf{Mfex}_*^a$-identifies $\mathcal{S}$. Let h, computable, be such that $(\forall f \in \mathcal{S})[\mathbf{M}(f) \leq h(\mathrm{MinProg}(f))]$. We assume without loss of generality that h is monotone increasing. We first construct an $\mathbf{M}'$ which on each $f \in \mathcal{S}$, converges to a "small" enough program, with "small" number of errors.

Let P range over finite sets of programs. By the s-m-n theorem (Rogers (1967)) there exists a recursive function Unify, mapping finite sets of programs to programs, such that $\varphi_{\mathrm{Unify}(\cdot)}$ can be defined as follows.

```
Begin Unify($P$)
On input $x$,
    search for $i \in P$ such that $\varphi_i(x)\downarrow$.
```

If and when such an i is found output $\varphi_i(x)$ (for the first such i found).

End

Let GOOD(f,k) be true iff the following three properties are satisfied.

(i) For each $i \in D_{\pi_2(k)}$, $\mathrm{card}(\{x \mid \varphi_i(x)\downarrow \neq f(x)\}) \leq a$.

(ii) For each $m \geq k$, $\mathbf{M}(f[m]) \in D_{\pi_2(k)}$.

(iii) $(\forall i \mid h(i) < \max(D_{\pi_2(k)}))(\exists x \leq k)[\varphi_i(x) \neq f(x)]$.

Note that, for $f \in \mathcal{S}$, there exists a k such that GOOD(f,k). Let $\mathrm{PROG}(f) = D_{\pi_2(\min(\{k \mid \mathrm{GOOD}(f,k)\}))}$. We construct $\mathbf{M}'$ such that, for all f, $\mathbf{M}'(f) = \mathrm{Unify}(\mathrm{PROG}(f))$.

Let MAYBEGOOD$(f[n],k)$ be true iff the following three properties are satisfied.

(i) For all $i \in D_{\pi_2(k)}$, $\mathrm{card}(\{x < n \mid \Phi_i(x) \leq n \wedge \varphi_i(x)\downarrow \neq f(x)\}) \leq a$.

(ii) For all m such that $k \leq m \leq n$, $\mathbf{M}(f[m]) \in D_{\pi_2(k)}$.

(iii) $(\forall i \mid h(i) < \max(D_{\pi_2(k)}))(\exists x \leq k)[\Phi_i(x) > n \vee \varphi_i(x) \neq f(x)]$.

Let $\mathbf{M}'(f[n]) = \mathrm{Unify}(D_{\pi_2(\min(\{n\} \cup \{k' \leq n \mid \mathrm{MAYBEGOOD}(f[n],k')\}))})$.
Let $h_1(j) = \max(\{\mathrm{Unify}(P) \mid P \subseteq \{x \mid x \leq h(j)\}\})$.

Claim 49 *For all $f \in \mathcal{S}$, there exists an i such that, $\mathbf{M}'(f)\downarrow = i$ and the following three conditions are satisfied.*

(i) $i \leq h_1(\mathrm{MinProg}(f))$.

(ii) $\mathrm{card}(\{x \mid \varphi_i(x)\uparrow\}) \leq a$.

(iii) $\mathrm{card}(\{x \mid \varphi_i(x)\downarrow \neq f(x)\}) \leq a * (1 + h(\mathrm{MinProg}(f)))$.

Proof. (i) holds since $\mathrm{PROG}(f) \subseteq \{j \mid j \leq h(\mathrm{MinProg}(f))\}$.

(ii) holds since the number of errors committed by the final programs, output by $\mathbf{M}$ on f, is bounded by a.

(iii) holds because $\mathrm{card}(\mathrm{PROG}(f)) \leq 1 + h(\mathrm{MinProg}(f))$ and for all $i \in \mathrm{PROG}(f)$, $\mathrm{card}(\{x \mid \varphi_i(x)\downarrow \neq f(x)\}) \leq a$ (by condition (i) in the definition of GOOD). ▮

Define mindiv, errset, patchset, patch, g as follows.

$$\mathrm{mindiv}(j,n) = \min(\{n\} \cup \{x \mid \Phi_j(x) \geq n\}).$$

$$\mathrm{errset}(j,n) = \{x < \mathrm{mindiv}(j,n) \mid \Phi_{\mathbf{M}'(\varphi_j[\mathrm{mindiv}(j,n)])}(x) \leq n \wedge \varphi_{\mathbf{M}'(\varphi_j[\mathrm{mindiv}(j,n)])}(x) \neq \varphi_j(x)\}.$$

$$\mathrm{patchset}(j,n) = \{\langle x, \varphi_j(x)\rangle \mid x \in \mathrm{errset}(j,n)\}.$$

$$\varphi_{\mathrm{patch}(l,D)}(x) = \begin{cases} y & \text{if } (\exists z)[\langle x,z\rangle \in D] \wedge [y = \min(\{z \mid \langle x,z\rangle \in D\})]; \\ \varphi_l(x) & \text{otherwise.} \end{cases}$$

$$g(j,n) = \begin{cases} \text{patch}(\mathbf{M}'(\varphi_j[m]), \text{patchset}(j,n)), & \text{if } \text{card}(\text{errset}(j,n)) \leq a \cdot [h(j)+1] \wedge \\ & \mathbf{M}'(\varphi_j[\text{mindiv}(j,n)]) \leq h_1(j), \\ & \text{and } m = \text{mindiv}(j,n); \\ 0, & \text{otherwise.} \end{cases}$$

mindiv(j,n) finds the minimal apparent divergent point for φ_j (using n as a complexity bound) and errset finds the set of apparent convergent errors (with respect to φ_j) in $\varphi_{\mathbf{M}(\varphi_j[\text{mindiv}(j,n)])}$. patchset contains the patches required for the errors in errset. Note that for $\varphi_j \in \mathcal{S}$, $\lim_{n\to\infty} g(j,n)$ is a program formed by patching the convergent errors in $\mathbf{M}'(\varphi_j)$.

It is easy to see that the following claim holds.

Claim 50 *For all j, if $\varphi_j \in \mathcal{S}$, then the following four properties are satisfied.*

(i) $\text{card}(\{g(j,n) \mid n \in N\}) \leq 1 + (1 + h_1(j)) * (1 + a * (1 + h(j)))$.

(ii) $\lim_{n\to\infty} g(j,n)\downarrow = k$ *such that,*

$\text{card}(\{x \mid \varphi_k(x)\uparrow\}) \leq a \wedge$
$\text{card}(\{x \mid \varphi_k(x)\downarrow \neq \varphi_j(x)\}) = 0$.

(iii) For all l, if $\varphi_l = \varphi_j$, then $\lim_{n\to\infty} g(l,n)\downarrow = \lim_{n\to\infty} g(j,n)$.

(iv) $\lim_{n\to\infty} g(j,n) = \text{patch}(\mathbf{M}'(\varphi_j), \{\langle x, \varphi_j(x)\rangle \mid \varphi_{\mathbf{M}'(\varphi_j)(x)\downarrow \neq \varphi_j(x)}\})$.

Let $\mathbf{M}''$ be such that, $\mathbf{M}''(f[n]) = \text{patch}(\mathbf{M}'(f[n]), \{\langle x, f(x)\rangle \mid x < n \wedge \Phi_{\mathbf{M}'(f[n])}(x) \leq n \wedge \varphi_{\mathbf{M}'(f[n])}(x) \neq f(x)\})$. It is easy to see that, for $\varphi_j \in \mathcal{S}$, $\mathbf{M}''(\varphi_j) = \text{patch}(\mathbf{M}'(\varphi_j), \{\langle x, \varphi_j(x)\rangle \mid \varphi_{\mathbf{M}'(\varphi_j)(x)\downarrow\neq\varphi_j(x)}\}) = \lim_{n\to\infty} g(j,n)$.

Thus from Theorem 3.3 in Chen (1982) (and using g, $\mathbf{M}''$ constructed above) we have that $\mathcal{S} \in \mathbf{Mex}^a$. ∎

Theorem 51 (Chen (1981)) $\mathbf{Mex}^* = \mathbf{Mfex}^*_*$.

Proof. Case and Smith (1983) showed that $\mathbf{Fex}^*_* = \mathbf{Ex}^*$. Chen (1981) showed that $\mathbf{Ex}^* = \mathbf{Mex}^*$. The theorem follows. ∎

5 Future Work

The results in section 3 (Theorems 18, 28, 38, 39 and 40) do not give the complete relationship (even for identification without anomalies) between the criteria of inference introduced. Methodologically there is a gap between the diagonalization and simulation results. Future work will concentrate on proving theorems which give the complete relationship between the inference criteria introduced. In particular we need to much more generally assess the effects of the presence of anomalies.

6 Acknowledgements

We would like to express our gratitude to Professor S. N. Maheshwari of the Department of Computer Science and Engineering at the Indian Institute of Technology, New Delhi for making the facilities of his department available to us during the preparation of this manuscript.

References

Angluin, D. (1980). Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, *21*, 46–62.

Barzdin, J. M. and Freivalds, R. (1974). Prediction and limiting synthesis of recursively enumerable classes of functions. *Latvijas Valsts Univ. Zimatm. Raksti*, *210*, 101–111.

Barzdin, J. M. and Podnieks, K. (1973). The theory of inductive inference. In *Mathematical Foundations of Computer Science*.

Blum, L. and Blum, M. (1975). Toward a mathematical theory of inductive inference. *Information and Control*, *28*, 125–155.

Blum, M. (1967). A machine independent theory of the complexity of recursive functions. *Journal of the ACM*, *14*, 322–336.

Case, J. (1974). Periodicity in generations of automata. *Mathematical Systems Theory*, *8*, 15–32.

Case, J. (1988). The power of vacillation. In Haussler, D. and Pitt, L. (Eds.), *Proceedings of the Workshop on Computational Learning Theory*, (pp. 133–142). Morgan Kaufmann Publishers, Inc. Expanded in Case (1992).

Case, J. (1992). The power of vacillation in language learning. Technical Report 93-08, University of Delaware. Expands on Case (1988); journal article under review.

Case, J., Jain, S., and Sharma, A. (1989). Convergence to nearly minimal size grammars by vacillating learning machines. In Rivest, R., Haussler, D., and Warmuth, M. (Eds.), *Proceedings of the Second Annual Workshop on Computational Learning Theory, Santa Cruz, California*, (pp. 189–199). Morgan Kaufmann Publishers, Inc.

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

Chen, K. (1981). *Tradeoffs in Machine Inductive Inference*. PhD thesis, SUNY at Buffalo.

Chen, K. (1982). Tradeoffs in inductive inference of nearly minimal sized programs. *Information and Control*, *52*, 68–86.

Daley, R. and Smith, C. (1986). On the complexity of inductive inference. *Information and Control*, *69*, 12–40.

Freivalds, R. (1975). Minimal Gödel numbers and their identification in the limit. *Lecture Notes in Computer Science*, *32*, 219–225.

Gold, E. M. (1967). Language identification in the limit. *Information and Control*, *10*, 447–474.

Gold, E. M. (1978). Complexity of automaton identification from given data. *Information and Control*, *37*, 302–320.

Jain, S. and Sharma, A. (1990). Program size restrictions in inductive learning. Technical Report 90-06, University of Delaware, Newark, Delaware.

Jain, S. and Sharma, A. (1991). Program size restrictions in inductive learning. In Powers, D. and Reeker, L. (Eds.), *Proceedings MLNLO'91, Machine Learning of Natural Language and Ontology, Stanford University, California*, (pp. 87–92). DFKI.

Kinber, E. B. (1974). On the synthesis in the limit of almost minimal Gödel numbers. *Theory Of Algorithms and Programs, LSU, Riga*, *1*, 221–223.

Kinber, E. B. (1977). On limit identification of minimal Gödel numbers for functions from enumerable classes. *Theory of Algorithms and Programs, Riga 1977*, *3*, 35–56.

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

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

Osherson, D. and Weinstein, S. (1982). Criteria of language learning. *Information and Control*, *52*, 123–138.

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

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

Rogers, H. (1958). Gödel numberings of partial recursive functions. *Journal of Symbolic Logic*, *23*, 331–341.

Rogers, H. (1967). *Theory of Recursive Functions and Effective Computability.* McGraw Hill, New York. Reprinted, MIT Press 1987.

Royer, J. (1987). *A Connotational Theory of Program Structure.* Lecture Notes in Computer Science 273. Springer Verlag.

Wiehagen, R. (1986). On the complexity of program synthesis from examples. *Electronische Informationverarbeitung und Kybernetik*, *22*, 305–323.

Zeugmann, T. (1983). On the synthesis of fastest programs in inductive inference. *Electronische Informationverarbeitung und Kybernetik*, *19*, 625–642.