Learning via Queries with Teams and Anomalies

by

William I. Gasarch[1]
Department of Computer Science
Institute for Advanced Computer Studies
The University of Maryland
College Park Maryland, 20742 USA

Efim B. Kinber
Computing Centre
Latvian State University
Riga, USSR

Mark G. Pleszkoch
Department of Computer Science
The University of Maryland
College Park Maryland, 20742 USA
and
IBM Application Solutions Division
Gaithersburg Maryland, USA

Carl H. Smith[2]
Department of Computer Science
Institute for Advanced Computer Studies
The University of Maryland
College Park Maryland, 20742 USA

Thomas Zeugmann
Department of Mathematics
Humboldt University
Berlin GDR

[1] Supported, in part, by National Science Foundation Grant CCR 8803641.
[2] Supported, in part, by National Science Foundation Grant CCR 8701104.

I. Introduction

Most work in the field of inductive inference regards the learning machine to be a passive recipient of data [5,6]. In [13] the passive approach was compared to an active form of learning where the machine is allowed to ask questions. In this paper we continue the study of machines that ask questions by comparing such machines to *teams* of passive machines [26]. This yields, via work of Pitt and Smith [19], a comparison of active learning with *probabilistic learning* [18]. Also considered are query inference machines that learn an *approximation* of what is desired. The approximation differs from the desired result in finitely many *anomalous* places. Passive approximate inductive inference has been extensively investigated [8,10,11,21,27].

The basic paradigm of asking questions has been applied to DNF formulas [1], CNF formulas [4], μ formulas [15], context-free grammars [2], deterministic one-counter automata [7], deterministic bottom up tree automata [23], deterministic skeletal automata [22], deterministic languages [16], and prolog programs [24]. Valiant also considered the issue briefly. [28]. For a nice summary of these results see [3].

Several intuitions about the use of queries for learning are implied by our results. Firstly, active learning machines can be simulated by a team of passive learning machines, but (often) not conversely. Secondly, the power of queries seems to be incomparable to that of allowing anomalies or BC-learning. Thirdly, there is often an infinite hierarchy of active learning based on mind changes, and another infinite hierarchy based on anomalies. Several of our results pertain to the quantifier structure of questions asked by learning machines. One of our results indicates that asking questions with more alternations of quantifiers leads to an increase in learning potential. Furthermore, the number of quantifiers is an important factor.

II. Notation and Definitions

Throughout this paper, φ_0, φ_1, φ_2, $\ldots$ denotes an acceptable programming system [17], also known as a Gödel numbering of the partial recursive functions [20]. We will say that program i computes the function φ_i. An (standard, passive) *inductive inference machine* (IIM) is a total algorithmic device that takes as input the graph of a recursive function (an ordered pair at a time) and outputs (from time to time) programs intended to compute the function whose graph serves as input [8,14]. (see Figure 1) An IIM M learns a recursive function f, if, when M is given the graph of f as input, the resultant sequence of outputs converges (after some point there are no more mind changes) to a program that computes f. In this case we write $f \in EX(M)$. The class EX is the collection of all sets $EX(M)$ (or subsets thereof) of functions learned by an IIM. If convergence is achieved after only c changes of conjecture we write $f \in EX_c(M)$, for $c \in \mathbb{N}$, where $\mathbb{N}$ denotes the natural numbers. The class of sets of functions identifiable by IIMs restricted to c mind changes is denoted by EX_c.

$$f(0), f(1), f(2) \ldots \longrightarrow \boxed{\text{IIM}} \longrightarrow p_0, p_1, p_2, \ldots$$

Figure 1.

The convergence criterion discussed above was *syntactic* in that convergence to a particular program was required. There is also a *semantic* convergence criterion whereby convergence is to a function. Specifically, we say that an IIM semantically converges iff almost all of the programs output compute the same function. In other words, convergence is to a sequence of (possibly syntactically different) programs all computing the same function. The resulting notion of learning is called BC for behaviorally correct.

In some cases, convergence (in either sense) to a program computing the input function *exactly* may not be required. Perhaps an *approximation* will do. If an IIM M, on input f, converges to a program that computes f everywhere, except on perhaps at most a *anomalous* inputs, we say that $f \in EX^a(M)$. The class EX^a is defined analogously to the definition of the class EX. A comparison of the classes EX_c^a arising from the consideration of various values for a and c appears in [10].

A collection, or team, of inductive inference machines, M_1, M_2, ..., M_n, infers a function f iff there is an i with $1 \leq i \leq n$ such that $f \in EX(M_i)$. In this case we write $f \in EX(M_1, \ldots, M_n)$. A set S of recursive functions is learned by the team iff each $f \in S$ is learned by *some* member of the team. Different member of the team will learn different members of S. If the team M_1, M_2, ..., M_n learns the set S, we write $S \in [1,n]EX$. The class $[1,n]EX$ is the collection of sets S that are inferrible by some team of n inductive inference machines. The definition of $[m,n]EX$, where m out of the n inference machines succeed can be found in [19]. The definition of the classes $[m,n]EX_c$ is analogous.

A *query inference machine* (QIM) is an algorithmic device that asks a teacher questions about some unknown function, and while doing so, outputs programs. The questions are formulated in some language L. Formally, a QIM is a total algorithmic device which, if the input is a string of bits $\vec{b}$, corresponding to the answers to previous queries, outputs an ordered pair consisting of a (possibly null) program e, called a *guess*, and a question ψ. (See Figure 2) Define two functions g (*guess*) and q (*query*) such that if $M(\vec{b}) = (p, \psi)$ then $g(M(\vec{b})) = p$ and $q(M(\vec{b})) = \psi$. Without loss of generality, we adopt the conventions that all questions are assumed to be in prenex normal form (quantifiers followed by a quantifier-free formula, called the matrix of the formula) and that questions containing quantifiers are assumed to begin with an existential quantifier. A QIM M learns a recursive function f if, when the teacher answers M's questions about f truthfully, the sequence of output programs converges to a program that computes f. In this case, we write $f \in QEX[L](M)$. For a fixed language L, the class $QEX[L]$ is the collection of all sets $QEX[L](M)$ for a QIM M. $QEX_c[L]$, $QEX^\star[L]$ and $QEX_c^\star[L]$ are defined similarly. Teams of query machines are defined analogously.

QIM	$\longrightarrow$	$\psi_1, \psi_2, \ldots$	(questions formulated in L)
	$\longleftarrow$	$b_1, b_2, \ldots$	(answers)
L	$\longrightarrow$	$p_1, p_2, \ldots$	(guesses)

Figure 2.

All the query languages that we will consider allow the use of quantifiers. Restricting the applications of quantifiers is a technique that we will use to regulate the expressive power of a query language. Of concern to us is the alternations between blocks of existential and universal quantifiers, as well as the total number of quantifiers. Suppose that $f \in QEX[L](M)$ for some M and L. If M only asks quantifier-free questions, then we will say that $f \in Q_0EX[L](M)$. If M only asks questions with existential quantifiers, then we will say that $f \in Q_1EX[L](M)$. In general, if M's questions begin with an existential quantifier and involve $d \geq 0$ alternations between blocks of universal and existential quantifiers, then we say that $f \in Q_{d+1}EX[L](M)$. Furthermore, if there are at most k quantifiers total in all blocks we say that $f \in Q^k_{d+1}EX[L](M)$. The classes $Q^k_dEX[L]$ and $Q^k_dEX_c[L]$ are defined analogously.

Now we introduce the languages that will be used. Every language allows the use of $\wedge$, $\neg$, $=$, $\forall$, $\exists$, symbols for the natural numbers (members of $\mathbb{N}$), variables that range over $\mathbb{N}$, and a single function symbol $\mathcal{F}$ which will be used to represent the function being learned. Inclusion of these symbols in every language will be implicit. The *base* language L contains only these symbols. If L has auxiliary symbols, then L is denoted just by these symbols. For example, the language that has auxiliary symbols for plus and less than is denoted by $[+, <]$. The language that has auxiliary symbols for plus and times is denoted by $[+, \times]$. The language with extra symbols for successor and less than is denoted by $[S, <]$, where S indicates the symbol for the successor operation. Such languages have "ϵ" as a symbol for "element of." The symbol "$\star$" will be used to denote an arbitrary language that includes all the symbols common to all the languages we consider and some (possibly empty) subset of recursive operators, e.g. $+$, $<$, $\times$ and S. Such a language will be called *reasonable*.

The following definitions are necessitated by our proof techniques. Suppose f is a function and n is a positive integer. For $j < n$, the j^{th} *n-ply* is the function $\lambda x[f(n \cdot x + j)]$. Clearly, any function can be determined from its n-plys. For any function f, let $I(f)$

denote the set of values y for which there are infinitely many x's with $f(x) = y$. For two functions f and g, we write $f =^{\star} g$ to mean that $f(x) = g(x)$ for all but finitely many x's. If $f(x) = g(x)$ except for at most a values of x, then we write $f =^{a} g$. If $f(x) = g(x)$ except for *exactly* a values of x, then we write $f =^{=a} g$.

III. Queries versus Teams

In this section we examine the simulation of active learning machines by teams of passive learning machines. Pitt [18] found an equvalence between teams of passive learning machines and probabilistic learning machines. Various trade offs with probability and other parameters of learning were investigated in [19].

THEOREM 1. $Q_{d+1}EX_0[\star] \subseteq Q_dEX[\star]$.

Proof: The $d = 0$ case is proven in [13]. Suppose $S \in Q_{d+1}EX_0[\star]$ as witnessed by the QIM M. We describe the operation of a QIM M' showing $S \in Q_dEX[\star]$. M' simulates M as follows. If M asks a question with at most d blocks of quantifiers, M' ask the same question and gives the answer to M. M' also outputs any conjecture produced by M during the simulation. After M produces its only conjecture, M' can stop the simulation. Suppose M asks a question ψ with $d+1$ blocks of quantifiers. To reduce notational complexity, we assume that the leftmost quantifier block contains a single quantifier. (The general case is similar.) Hence, ψ looks like $\exists x \forall \ldots \phi(x, y_1, \ldots, y_n)$ for some n and quantifier free ϕ. Consider the following questions (with d blocks of quantifiers).

$$\psi_0 = \phi(0, y_1, \ldots, y_n)$$

$$\psi_1 = \phi(1, y_1, \ldots, y_n)$$

$$\vdots$$

If the correct answer to ψ is "NO", then the correct answer to each of ψ_0, ψ_1, ... is also "NO". However, if the correct answer of ψ is "YES," then there will be a j such that the correct answer to ψ_j is "YES." In simulating M, M' answers ψ as NO and simultaneously continues the simulation *and* while doing so asks ψ_0, ψ_1, … until, if ever a j is found such that ψ_j is answered "YES." If such a j is found, the simulation of M is restarted from the beginning, only this time when M asks ψ, M' provides the answer "YES."

During the course of M''s operation, it may be working on several questions ψ at one time. The number of such questions will be finite as M can only ask finitely many questions before outputting its only conjecture. Hence, M''s simulation of M can be restarted only finitely often. In the case ψ starts with a block of universal quantifiers, the roles of "YES" and "NO" are reversed. Blocks of more than one quantifier are handled by considering vectors of values instead of x in the ψ_i's. ☒

THEOREM 2. $\forall c, d \in \mathbb{N}$, $Q_{d+1}EX_c[\star] \subseteq [1, c+1]Q_dEX[\star]$.

Proof: Let c and d be given. Suppose M is a QIM making at most c mindchanges that witnesses $S \in Q_{d+1}EX_c[\star]$. M is simulated by a team M_0, …, M_c where M_i $(0 \leq i \leq c)$ simulates M asking the same questions and providing M with the correct answers. However, instead of faithfully reproducing M's conjectures, M_i ignores all but the $i+1^{\text{st}}$ conjecture which is used as M_i's only output. ☒

The question of whether or not the inclusion of Theorem 2 is proper naturally arises. The answer depends on the query language.

THEOREM 3. For all $c \in \mathbb{N}$, $Q_1EX_c[S, <] \subset [1, c+1]EX$.

Proof: By the $d = 0$ case of Theorem 2, $Q_1EX_c[S, <] \subseteq [1, c+1]EX$ and by Theorem 10 of [13] $EX - QEX_c[S, <] \neq \emptyset$. Hence, the theorem follows. ☒

Next, we examine the language $[+, <]$. Not only do we compare the appropriate query inference classes with team inference classes, we are also able to answer a problem left open in [13]. The solution to this problem yields another level in a suspected infinite hierarchy based on alternation of quantifiers in the query language.

THEOREM 4. There is a $T \in (EX_0^1 \cap Q_2EX_1[<]) - Q_1EX[+,<]$.

Proof: Recall that $I(f)$ denotes the set of values that appear infinitely often in the range of f. Let T be defined as follows:

$$\begin{aligned} T = & \{f \mid \varphi_{f(0)} = f \text{ and } I(f) = \emptyset\} \cup \\ & \{f \mid \varphi_{f(0)} =^1 f, I(f) = \{e\}, \text{ and} \\ & \qquad \forall x > \mu z[f(z) = e], \varphi_{f(0)}(x) = f(x) \text{ and} \\ & \qquad \forall x, y > \mu z[f(z) = e], (f(x) \neq e \text{ and } x \neq y) \Rightarrow f(x) \neq f(y)\}. \end{aligned}$$

$T \in EX_0^1$ is witnessed by the IIM that always outputs as its only conjecture the value $f(0)$. The QIM that witnesses that $T \in Q_2EX_1[<]$ first finds the value of $f(0)$ by asking $\mathcal{F}(0) = 0$? $\mathcal{F}(0) = 1$? $\cdots$. The QIM then outputs $f(0)$. Then ask the following questions until (if ever) the unique $e \in I(f)$ is found:

$$\forall x \exists y [y > x \text{ and } \mathcal{F}(y) = 0]?$$

$$\forall x \exists y [y > x \text{ and } \mathcal{F}(y) = 1]?$$

$$\vdots$$

A "YES" answer indicates that $I(f)$ is not empty and an appropriate patched version of program $f(0)$ is output.

The proof that $T \notin Q_1EX[+,<]$ is a highly nontrivial modification of the proof that the set of recursive functions is not in $Q_1EX[+,<]$ from [13] (Theorem 13). Let M be a QIM that asks questions using the query language $[+,<]$ restricted to sentences with only existential quantifiers. We construct a recursive function $f \in T$ in effective stages of finite extension. The function f will be computed by program e described below, e.g. $f = \varphi_e$. The finite amount of f determined prior to stage s is denoted by f^s. The least number not in the domain of f^s is denoted by x^s. By way of initialization, by implicit use of the recursion theorem, $f^0 = \{(0, e)\}$. The function f is determined by the execution of the following stages in their natural order.

Begin stage s. Suppose that so far in the construction M has asked m questions and received answers b_1, b_2, …, b_m. Let $j = g(M(b_1 b_2 \dots b_m))$, M's most recent guess. As in other diagonalization arguments in inductive inference, we simultaneously look to make the current guess wrong or force a mind change. In addition we extend the function except at the point we are trying to diagonalize against. If stage s does not terminate, then f will be defined everywhere except on a single point. In this case, a patched version of f will suffice to obtain the desired result.

The search for an extension forcing a mind change will involve M asking more questions. *Several* questions may have to be answered during stage s before an extension forcing a mind change will be found. This will also involve fixing certain extensions to f^s that must be used in the event that the mind change is *not* found before a diagonalization point is found. Let $\vec{b} = \langle b_1, \dots, b_m \rangle$. This vector of responses will be lengthened during stage s. To reduce notation, the various, larger and larger, vectors will not be indexed. Consequently, $\vec{b}$ always denotes the *current* vector. Similarly, let σ denote the *current* fixed portion of f. Initialize $\sigma = f^s$ so that it will always be the case that $f^s \subseteq \sigma \subseteq f^{s+1}$. Simultaneously execute the following two substages.

Substage 1. Diagonalize against the current guess. If $\varphi_j(x^s)$ converges before a mind change is found in substage 2, then set $f^{s+1} = \sigma \cup \{(x^s, 1 \dot{-} \varphi_j(x^s))\}$ and go to stage $s+1$.

Substage 2. Force a mind change or extend f^s. Let y be the least number not in the range σ. In trying to answer questions, we will assume $f(x^s) = y$, although $f(x^s)$ will remain undefined. Let $\psi = q(M(\vec{b}))$, M's most recent query. Define $\psi_0 = \psi$ and $\psi_1 = \neg\psi$. For $i \in \{0, 1\}$, use Lemma 4 of [13] to effectively find out if there is a finite sequence τ_i (with no repeated values) extending $\sigma \cup \{(x^s, y)\}$ such that any function extending τ_i will make ψ_i true. By Lemma 3 of [13], for some $i \in \{0, 1\}$, τ_i exists.

If there exists $i \in \{0, 1\}$ such that $g(M(\vec{b}i)) \neq j$ and τ_i exists then choose the least such i and the least such τ_i and set $f^{s+1} = \tau_i$, $\vec{b} = \vec{b}i$, and go to stage $s+1$, (this forces M to change its mind at stage $s+1$)

else let i be the least number such that τ_i exists, set $\vec{b} = \vec{b}i$, $\sigma = \sigma \cup \tau_i \cup \{(z, y)\} - \{(x^s, y)\}$, for z the least number not in the domain of τ_i, and repeat substage 2 for these new values of σ and $\vec{b}$.

End stage s.

If every stage of the construction terminates, then $f(x)$ is defined for all x and $I(f) = \emptyset$, hence $\varphi_{f(0)} = f$, so $f \in T$. If M converges to j when trying to infer f, then, by the failure of substage 2 to extend f past some point, φ_e (since it is extended by substage 1 infinitely often) is wrong on infinitely many arguments. Hence, $f \notin Q_1EX[+,<](M)$.

If some stage s never terminates, then substage 2 is executed infinitely often. Consequently, f eventually becomes defined on every argument except x^s. We show that M does not infer $h = f \cup \{(x^s, y)\}$. Let $\vec{b}$ denote the value of $\vec{b}$ on entry into stage s and let $j = g(M(\vec{b}))$. By the failure of substage 2 to terminate stage s, when M tries to infer h, it will converge to j. By the failure of substage 1 to terminate stage s, $\varphi_j(x^s)$ is undefined. Hence, $h \notin Q_1EX[+,<](M)$.

$\boxtimes$

The class EX_0^1 is very restrictive. As a consequence of known inclusions [10,26], the set T from Theorem 4 is also contained in the class $[1,2]EX$, and all its supersets. A question that naturally arises is whether or not Theorem 4 can be modified to consider the class $Q_2EX[\ ]$ instead of $Q_2EX_1[<]$.

THEOREM 5. $Q_2EX[\ ] - Q_1EX[+,<] \neq \emptyset$.

Proof: Let

$$S = \{f \mid \varphi_{f(0)} = f \text{ and } \neg(\exists k, l \forall x > l\ \exists y_1, \ldots, y_k) f(x) = f(y_1) = \ldots = f(y_k)\}$$

$$\cup \{f \mid \varphi_{f(0)} = f \text{ and } (\exists k, l \forall x > l\ \exists y_1, \ldots, y_k) f(x) = f(y_1) = \ldots = f(y_k) \text{ and } \varphi_l = f \text{ for } l \text{ least}\}$$

The theorem follows using techniques similar to the proof of Theorem 4. $\boxtimes$

COROLLARY 6. For all $c \in \mathbb{N}$, $Q_1EX_c[+,<] \subset [1,c+1]EX$.

Proof: By Theorem 2 with $d = 0$ and $\star =$"$+, <$," $Q_1EX_c[+,<] \subseteq [1,c+1]EX$. By Theorem 4, $EX_0^1 - Q_1EX[+,<] \neq \emptyset$. Since $EX_0^1 \subseteq EX \subseteq [1,c+1]EX$ [26] the corollary follows. ☒

COROLLARY 7. $Q_1EX[+,<] \subset Q_2EX[+,<]$.

Proof: The inclusion holds by definition. We show that the inclusion is proper. By Theorem 4, $Q_2EX_1[<] - Q_1EX[+,<] \neq \emptyset$. Since $Q_2EX_1[<] \subseteq Q_2EX[+,<]$, the corollary follows. ☒

IV. Queries versus BC

In this section we compare active learning with BC learning. As a corollary to the main theorem in this section we obtain an infinite hierarchy based on mind changes.

THEOREM 8. For all $c \in \mathbb{N} - \{0\}$, $Q_1EX_c[S] - [1,c]BC \neq \emptyset$.

Proof: For n a positive integer, $j < n$, and f a function, let f_j^n denote the j^{th} n-ply of f.

Let c be given. Let S_c be defined as follows:

$$
\begin{aligned}
S_c = \bigcup_{1 \leq i \leq c-1} \{ f \,|\, \exists e_1, e_2, \ldots, e_i \\
& f_0^{c+1} = \lambda x[e_1] \\
& f_1^{c+1} =^{\star} \lambda x[e_2] \\
& \vdots \\
& f_{i-1}^{c+1} =^{\star} \lambda x[e_i] \\
& f_j^{c+1} \neq^{\star} \lambda x[k], k \in \mathbb{N}, i \leq j \leq c \\
& f_j^{c+1}(x) \neq e_i, 1 \leq j \leq c, x \in \mathbb{N} \\
& f = \varphi_{e_i} \} \\
\cup \{ f \,|\, \exists e_1, e_2, \ldots, e_c \\
& f_0^{c+1} = \lambda x[e_1] \\
& f_1^{c+1} =^{\star} \lambda x[e_2] \\
& \vdots \\
& f_{c-1}^{c+1} =^{\star} \lambda x[e_c] \\
& f = \varphi_{e_c} \text{ or } f_c^{c+1} =^{\star} \lambda x[e_c] \}
\end{aligned}
$$

A QIM that witnesses $S_c \in Q_1EX_c[S]$ behaves as follows. The value of e_1 can easily be found. Output this value. To find e_2 (if it exists) ask the following question for all values of a and b:

$$\forall y[(y \neq 0 \text{ and } y \neq 1 \text{ and } \ldots \text{ and } y \neq a) \Rightarrow (\mathcal{F}(y) = e_1 \Rightarrow \mathcal{F}(y+1) = b)]?$$

When a "YES" answer is received, then $e_2 = b$. Output e_2, if it exists. By a similar process, find and output e_3, e_4, …, e_c. If e_c is output, we must also check to see if $f_c^{c+1} =^{\star} \lambda x[e_c]$.

The proof is completed by showing that $S \notin [1, c]BC$. Suppose M_1, …, M_c are IIMs. Using the operator recursion theorem [9] an infinite monotone increasing sequence of programs, $p(0)$, $p(1)$, … is constructed, in stages, such that $\varphi_{p(i)}$ will be in $S - (BC(M_1) \cup$

$\cdots \cup BC(M_c))$, for some i. Program $p(0)$ will start with $M_1, \ldots, M_c$ on a queue in that order. The program proceeds by trying to diagonalize, using standard techniques, against the IIM at the front of the queue. If successful, program $p(0)$ moves the IIM at the front of the queue to the rear and goes to the next stage. Continuing in this fashion, $p(0)$ will diagonalize against each of $M_1, \ldots, M_c$ infinitely often in a round robin manner. If the IIM at the front of the queue has converged on the segment of $\varphi_{p(0)}$ determined so far to a program for a finite function, then $p(0)$'s search for a diagonalization point will fail. Another one of the $p(i)$'s will continue at that point, in an identical fashion, except that the IIM at the front of $p(0)$'s queue will have been permanently removed. We say that program $p(0)$ is at *level* 1 and that the $p(i)$ with only $c-1$ IIMs on its queue is at level 2. The program called $p(i)$ may also be unable to find a diagonalization point for similar reasons. Consequently, there will be other programs in the sequence at levels 3, 4, ..., $c+1$. The program operating at level k will have $(c+1)-k$ IIMs on its queue.

It is possible that the program at level k will succeed only ofter the program at level $k+1$ has been successful in finding extensions. In this case, a new program must be started at level $k+1$, extending the recently revised program at level k. This new program will be the "next" $p(i)$, i.e. $p(i)$ where i is least such that $p(i)$ has not yet been mentioned, explicitly or implicitly, in the construction. Programs at level k will be explicitly activated and deactivated by programs at levels $1, \ldots, k-1$. e_k $(1 \leq k \leq c+1)$ will denote the currently active program (one of the $p(i)$'s) at level k. The construction will insure that $\varphi_{e_1} \subseteq \varphi_{e_2} \subseteq \ldots \subseteq \varphi_{e_{c+1}}$. The finite initial segment of φ_{e_k} determined prior to stage s of the construction of the current e_k will be denoted by σ_k^s.

The initial configuration of the construction is as follows. For $1 \leq k \leq c$, $e_k = p(k-1)$ with queue $M_k, \ldots M_c$. $e_{c+1} = p(c)$ with an empty queue. For $1 \leq k \leq c+1$, $\sigma_k^0 = \emptyset$. Program e_{c+1} will be determined in its entirely at its activation. Program e_1 $(p(0))$ can never be deactivated. In order to avoid a notational nightmare, we present the construction for the $c = 2$ case only. This construction has three levels. We will be concerned with the zeroth, first and second 3 plies of the functions we constructed. To simplify notation, these will be referred to as the 0^{th}, 1^{st} and 2^{nd} plies, respectively.

Stage s in the simultaneous construction of e_1, e_2 and e_3.

Suppose the queue is M, M' in front to rear order. Simultaneously perform the following 2 steps.

Step 1: Make another one of M's guesses wrong.

Look for a ρ and a τ such that

T1.1 $\tau \supseteq \rho \supset \sigma_1^s$ such that:

T1.2 If x is in domain $(\tau - \sigma_1^s)$ and x is on the 0^{th} ply of τ then $\tau(x) = e_1$, and

T1.3 If x is in domain $(\tau - \sigma_1^s)$ and x is on the 1^{st} or 2^{nd} ply of τ then $\tau(x) \in \{e_2, e_2 + 1\}$, and

T1.4 Domain of ρ and τ are initial segments of the integers, and

T1.5 $\varphi_{M(\rho)}(x) \neq \tau(x)$ for some x in domain $(\tau - \sigma_1^s)$.

If such a τ is found then perform the following actions:

A1.1 Stop work on Step 2, and

A1.2 Set $\sigma_1^{s+1} = \tau$, and

A1.3 Move M to the rear of the queue, and

A1.4 Deactivate e_2 and e_3, and

A1.5 Activate a new e_2 with $\sigma_2^{s+1} = \tau$, and

A1.6 Go to stage $s + 1$.

Step 2: Give up on M, try M'.

Execute the following Substages in their natural order. $\sigma_2^{s,t}$ denotes the finite initial segment of φ_{e_2} (for the current e_2) determined prior to Substage t. $\sigma_2^{s,0} = \sigma_1^s$.

Begin Substage t. Activate a new e_3 and define:

$$\varphi_{e_3}(x) = \begin{cases} \sigma_2^{s,t}(x) & \text{if } x \in \text{domain } \sigma_2^{s,t}, \\ e_1 & \text{if } x \text{ is on the } 1^{\text{th}} \text{ ply and } x \notin \text{domain } \sigma_2^{s,t}, \\ e_2 & \text{otherwise.} \end{cases}$$

Try and make another one of M''s guesses wrong by searching for

a ρ and a τ such that:

T2.1 $\tau \supseteq \rho \supset \sigma_2^{s,t}$

T2.2 If x is in domain $(\tau - \sigma_2^{s,t})$ and x is on the 0^{th} ply of τ then $\tau(x) = e_1$, and

T2.3 If x is in domain $(\tau - \sigma_2^{s,t})$ and x is on the 1^{st} ply of τ then $\tau(x) = e_2$, and

T2.4 If x is in domain $(\tau - \sigma_2^{s,t})$ and x is on the 2^{nd} ply of τ then $\tau(x) \in \{e_2, e_2 + 1\}$, and

T2.5 There are x and y in domain $(\tau - \sigma_2^{s,t})$ such that $\tau(x) = e_2$ and $\tau(y) = e_2 + 1$, and

T2.6 Domain of ρ and τ are initial segments of the integers, and

T2.7 $\varphi_{M'(\rho)}(x) \neq \tau(x)$ for some x in domain $(\tau - \sigma_2^{s,t})$.

If such a τ is found then perform the following actions:

A3.1 Set $\sigma_2^{s,t+1} = \tau$, and

A3.2 Deactivate e_3, and

A3.3 Go to Substage $t + 1$.

End Substage t.

End Stage s.

Case 1. Every stage s *terminates.* Since e_1 is never deactivated, $e_1 = p(0)$ throughout the construction. Let $f = \varphi_{e_1}$. f is a recursive function since every stage defines f on a larger initial segment of its domain. By T1.2, $f_0^3 = \lambda x[e_1]$. By actions A1.4, and A1.5, e_2 is deactivated and reactivated with a new index at every stage. By T1.3, neither the first nor second plies of f are finite variants of constant functions. Only values of various e_2's and their successors are placed in the range of f along the second and third plies. By the monotinicity condition of the operator recursion theorem, all these values will be larger than e_1. Hence, $f \in S_2$ by the $i = 1$ case of the first clause of the definition of S_2.

Choose $M \in \{M_1, M_2\}$. M is at the front of the queue at the beginning of infinitely many stages. Actually, for the $c = 2$ case that we are doing, M will be at the front of queue at every other stage. At each such stage there is a different ρ and an x such that $\rho \subset f$ and $\varphi_{M(\rho)}(x) \neq f(x)$. Hence, infinitely often, M, on input f, outputs an incorrect program. Consequently, $f \notin BC(M)$. Since M was chosen arbitrarily, $f \in (S_2 - [1,2]BC)$.

Case 2. Some stage s never terminates. Let s be the least such stage. Suppose the queue at the beginning of stage s is M, M' in front to rear order. Program e_2 is not deactivated at or past stage s since if action A1.4 is executed, so will action A1.6 and stage s will terminate. Every $\tau \supset \sigma_1^s$ that is an initial segment of φ_{e_2} or of φ_{e_3} for any e_3 active during stage s will be considered in T1.1 through T1.4. Program $M(\tau)$, for each such τ, computes a finite function, as otherwise a ρ and a τ satisfying T1.1 through T1.5 would be found and stage s would terminate.

Case 2.1. Every substage t terminates. Let $f = \varphi_{e_2}$. f is a recursive function since every substage defines f on a larger initial segment of its domain. By T2.2, $f_0^3 = \lambda x[e_1]$. By T2.3, $f_2^3(x) = e_1$, for all $x \notin$ domain σ_1^s. T2.5 insures that the third ply of f is not a finite variant of a constant function. Only values e_2 and $e_2 + 1$ are placed in the range of f along the third ply. By the monotinicity condition of the operator recursion theorem these values will be larger than e_1. Hence, $f \in S_2$ by the second clause of the definition of S_2.

By the remarks in the beginning of Case 2, M fails to BC identify f. At each substage t there is a different ρ and an x such that $\rho \subset f$ and $\varphi_{M'(\rho)}(x) \neq f(x)$. Hence, infinitely often, M', on input f, outputs an incorrect program. Consequently, $f \notin BC(M')$. Since M and M' we chosen arbitrarily, $f \in (S_2 - [1,2]BC)$.

Case 2.2. Some substage t never terminates. Let t be the least such stage. Let $f = \varphi_{e_3}$ as defined at the beginning of substage t. Since $f_2^3 =^\star \lambda x[e_2]$, $f \in S_2$ by the second clause of the definition of S_2. Every τ that is an initial segment of f will be considered in T2.1 through T2.6. Program $M'(\tau)$, for each such τ, computes a finite function, as otherwise a ρ and a τ satisfying T2.1 through T2.6 would be found and substage t would terminate. Hence, M' cannot BC identify f. By the remarks in the beginning of Case 2, M fails to BC identify f. Since M and M' we chosen arbitrarily, $f \in (S_2 - [1,2]BC)$. ☒

The above argument can be modified to show the following.

THEOREM 9. For all $c \in \mathbb{N}$, $Q_1EX_c[S] - \cup_{a \in \mathbb{N}}[1,c]BC^a \neq \emptyset$.

Proof: Choose $a \in \mathbb{N}$. Modify the proof of Theorem 8 as follows. Instead of looking for an x in T1.5 and T2.7 look for $a+1$ distinct such x's. In Case 1 and 2.1, instead of obtaining M (or M') infinitely often outputting a wrong program, we now have M (or M') infinitely often outputting a program that is wrong in at least $a+1$ places. $\boxtimes$

COROLLARY 10. BC and $Q_1EX_1[S,<]$ are incomparable.

Proof: By Theorem 8, $Q_1EX_1^*[S,<] - BC \neq \emptyset$. In [13] it is shown that $EX - Q_1EX_1[S,<] \neq \emptyset$. $\boxtimes$

The following corollary of Theorem 8 yields a multitude of infinite hierarchies based on the number of mind changes allowed a QIM asking questions involving a single type of quantifier. There is a hierarchy for each language.

COROLLARY 11. Let L be any language that contains a symbol S for successor. Then for each $c \in \mathbb{N}$, $Q_1EX_c[L] \subset Q_1EX_{c+1}[L]$.

Proof: The inclusion holds by definition, we show that it is proper. Let $c \in \mathbb{N}$ be given. By the $d = 0$ case of Theorem 2, $Q_1EX_c[L] \subseteq [1,c+1]Q_0EX[L]$. Since each QIM that asks quantifierless questions can be replaced by an equipowerful IIM [13], $[1,c+1]Q_0EX[L] = [1,c+1]EX$. ¿From the results in [26] it follows that $[1,c+1]EX \subset [1,c+1]BC$. Hence, $Q_1EX_c[L] \subset [1,c+1]BC$. By Theorem 8, there is an element of $Q_1EX_{c+1}[L]$ that is not in $[1,c+1]BC$. The corollary follows. $\boxtimes$

THEOREM 12. $Q_2EX_0[S,<]$ and $\cup_{c=1}^{\infty}[1,c]BC$ are incomparable.

Proof: The set of primitive recursive functions is in EX, and consequently, in BC. In [13] it was shown that the primitive recursive functions were not in $Q_2EX_0[S,<]$. To complete the proof, it suffices to find a set T such that $T \in Q_2EX_0[S,<] - \cup_{c=1}^{\infty}[1,c]BC$. Let

$$T = \bigcup_{c=1}^{\infty} S_c$$

where S_c is as defined in the proof of Theorem 8. By Theorem 8, $T \notin \cup_{c=1}^{\infty}[1,c]BC$.

A QIM that witnesses $T \in Q_2EX_0[S,<]$ behaves as follows. First find a $c > 0$ such that $f(0) = f(c)$. Then the function serving as input is from S_c. Furthermore, the value of $f(0)$ is e_1. To find out if e_2 exists, ask:

$$\exists y, z \forall x[x > y \text{ and } \mathcal{F}(x) = e_1 \Rightarrow \mathcal{F}(x+1) = z]?$$

If e_2 exists, z is its value. Continue in this fashion to find e_3 (if it exists) by asking:

$$\exists y, z \forall x[x > y \text{ and } \mathcal{F}(x) = e_2 \Rightarrow \mathcal{F}(x+1) = z]?$$

☒

V. Queries versus Anomalies

A further analysis of the proof of the result that $Q_0EX[\star] = EX$ from [13] immediately yields the following.

THEOREM 13. For all $a \in \mathbb{N} \cup \{\star\}$, for all $c \in \mathbb{N}$, $Q_0EX_c^a[\star] = EX_c^a$.

The next result shows a completely different outcome when questions using a single quantifier are allowed.

THEOREM 14. For all $a \in \mathbb{N} \cup \{\star\}$, $Q_1^1 EX_0[\,] - \cup_{c \geq 0} EX_c^a \neq \emptyset$.

Proof: Pick $a \in \mathbb{N} \cup \{\star\}$. First we define T to be the set of *step functions*. T contains all and only the functions f such that either $f = \lambda x[0]$ or there are constants $c_1, \ldots, c_n$, for some $n \in \mathbb{N}$ such that:

$$f(x) = \begin{cases} 0 & \text{if } x < c_1, \\ 1 & \text{if } c_1 \leq x < c_2, \\ \vdots & \\ c_n & \text{if } c_n \leq x. \end{cases}$$

In [10] it is shown that T is not contained in $\cup_{c \geq 0} EX_c^a$. The proof is completed by presenting a QIM M that $Q_1^1 EX_0$ infers T using a query language without special symbols. First, M determines if the input function is the everywhere zero function by asking:

$$\forall x[\mathcal{F}(x) = 0]?$$

If the answer is "YES" then M outputs a program for the constant zero function and stops. If the answer is "NO" M continues by trying to find c_1 by asking:

$$\mathcal{F}(0) = 1?$$

$$\mathcal{F}(0) = 0 \wedge \mathcal{F}(1) = 1?$$

$$\mathcal{F}(0) = 0 \wedge \mathcal{F}(1) = 0 \wedge \mathcal{F}(2) = 1?$$

$$\mathcal{F}(0) = 0 \wedge \mathcal{F}(1) = 0 \wedge \mathcal{F}(2) = 0 \wedge \mathcal{F}(3) = 0?$$

If $f \in T$, then eventually a "YES" answer will be the response to one of the above questions. When the "YES" answer arrives, the value of c_1 is known. M then asks:

$$\forall x[(x < c_1 \Rightarrow \mathcal{F}(x) = 0) \wedge x \geq c_1 \Rightarrow \mathcal{F}(x) = 1)]?$$

If the answer is "YES" then M outputs a program for the following function g:

$$g(x) = \begin{cases} 0 & \text{if } x < c_1, \\ 1 & \text{otherwise.} \end{cases}$$

If the answer is "NO" M searches for c_2 in a manner similar to the method for finding c_1. Again, the search will succeed iff $f \in T$. The process continues until M has found all the relevant constants $c_1, \ldots, c_n$. Any function in T will have such constants for some n. The questions using the universal quantifier are used to determine if the *last* such constant has just been found. With the appropriate constants in hand, M easily outputs the correct program. A program is output only when all the proper constants have been found. M outputs at most one program. $\boxtimes$

This last theorem shows that $Q_1^1 EX[\star] - EX_c \neq \emptyset$, for all c. Hence the result $Q_1 EX_0[\star] \subseteq EX$ from [13] cannot be improved by substituting EX_c for EX. It is also possible to prove Theorem 14 using, instead of T, a set S of recursive functions that are almost every where $\{0, 1\}$ valued and have the property that the largest value that does not map to a 0 or a 1 is an index for the function. The proof becomes more involved with the use of a more complicated set. Wiehagen [29] has shown that this set S cannot be inferred by any *consistent* inductive inference machine (i.e., machines that only output conjectures that agree with all the data seen so far), thereby ruling out a potential strengthening of the result that $Q_1 EX_0[\star] \subseteq EX$ to a more restrictive class than EX. An inductive inference machine is consistent if all its conjectures are consistent with all the data used to make the conjecture. This contrasts with the fact that any class in EX_0 can be inferred consistently. Moreover, the class S contains arbitrarily complex functions since suitable finite variants of arbitrarily complex functions are in S and some these are also arbitrarily complex [25]. Hence, a QIM that asks only single quantifier questions can learn a set of arbitrarily complex functions that no passive IIM can learn when restricted to a fixed finite number of mind changes. Finally, the class S cannot be *reliable* [8] identified. For reliable inference, convergence is synonymous with identification. Hence, another possible strengthening of our results is ruled out.

Now we consider whether or not asking questions enables the precise (no anomalies) learning of functions that, without queries, can only be learned by IIMs that tolerate anomalies. As we shall see, not even a *single* error can always be corrected. A strengthening of Theorem 4 is easily obtainable.

THEOREM 15. For all $a \in \mathbb{N}$, there is a set $T^a \in (EX_0^{a+1} \cap Q_2EX_1[<]) - Q_1EX^a[+,<]$.

Proof: Define T^a as follows:

$$
\begin{aligned}
T^a = \{f \mid & \varphi_{f(0)} = f \text{ and } I(f) = \emptyset\} \cup \\
& \{f \mid \varphi_{f(0)} =^{a+1} f, I(f) = \{e\}, \text{ and} \\
& \quad \forall x > \mu z[f(z) = e], \varphi_{f(0)}(x) = f(x) \text{ and} \\
& \quad \forall x, y > \mu z[f(z) = e], x \neq y \text{ and } f(x) \neq e \Rightarrow f(x) \neq f(y)\}.
\end{aligned}
$$

An easy modification of the proof of Theorem 4 , and a proof that $T^a \in Q_2EX_1[<]$ now suffices. ☒

COROLLARY 16. For all $a \in \mathbb{N}$, $EX_0^{a+1} - Q_1EX^a[+,<] \neq \emptyset$.

Proof: Immediate from Theorem 15. ☒

This result yields the following hierarchy:

$$Q_1EX[+,<] \subset Q_1EX^1[+,<] \subset \cdots \subset Q_1EX^a[+,<] \subset Q_1EX^{a+1}[+,<] \subset \cdots$$

COROLLARY 17. For all $a \in \mathbb{N}$, the classes EX^{a+1} and $Q_1EX^a[+,<]$ are incomparable.

Proof: By Corollary 16, it suffices to show that $Q_1EX^a[+,<] - EX^{a+1} \neq \emptyset$. Since $EX^{a+1} \subset BC$ [10] it suffices to show that $Q_1EX[+,<] - BC \neq \emptyset$. This is just the $c = 0$ case of Theorem 8. ☒

VI. Blocks of Quantifiers

In this section we establish a hierarchy based on the total number of quantifiers a QIM is allowed to use in phrasing its questions. Many results is logic and theoretical computer science suggest that increasing the number of alternations of quantifiers (number of blocks) would increase the capabilities of the resultant QIM. This expectation is reflected in Corollary 7. Below we present a *finer* approach. We establish hierarchies based on the total number of quantifiers *all of the same type*.

In order to carry out our next construction, we need to examine the expressive power of statements with k existential quantifiers in $[+, <]$. The following mathematical lemma concerns the expressive power of k existential quantifiers in $[+, \times, <]$. Although we do not need the additional strerngth of the lemma, it is no more difficult to prove and it may be useful at some later point. Basically, the following lemma says that k existential quantifiers are not sufficient to ask the question, "Is there some value in the range of $\mathcal{F}$ that occurs at least $k+1$ times?"

LEMMA 18. Let θ be a query in $[+, \times, <]$ with at most k existential quantifiers, let $m \in \mathbb{N}$, and let σ be a finite function with domain an initial segment of $\mathbb{N}$ such that any recursive function f extending σ satisfying the following two properties makes θ false.

1. if x is in the domain of the f and not in the domain of σ then $f(x) \geq m$.
2. no value appears in range$(f)-$ range(σ) more than k times.

Then there exists an $m' \geq m$ and a finite function τ such that the domains of τ and σ are disjoint and some value occurs in the range of τ at least $k+1$ times, and any function f extending $\sigma \cup \tau$ using values $\geq m'$ at most k times makes θ false.

Proof: Let $\theta = \exists x_1 \cdots \exists x_k \psi(x_1, \ldots, x_k)$. Let G_0 be the terms in $\psi(x_1, \ldots, x_k)$ which occur as arguments to the function symbol $\mathcal{F}$. For example, if θ is

$$\exists x \exists y\, (\mathcal{F}(3) = \mathcal{F}((x \times \mathcal{F}(y)) + \mathcal{F}(18))) \wedge (x > 4) \wedge (\mathcal{F}((x \times y) + x + x) = 52),$$

then

$$G_0 = \{3, (x \times \mathcal{F}(y)) + \mathcal{F}(18), y, 18, (x \times y) + x + x\}.$$

For a given finite function σ, let G_σ be the collections of terms formed by taking elements of G_0 and substituting (in all possible ways) members of the range of σ for the value of $\mathcal{F}$. For example, given G_0 as above and $\sigma = \{(0,5),(1,7)\}$,

$$\begin{aligned} G_\sigma =& \{3, (x \times 5)+5, (x \times 5)+7, (x \times 7)+5, (x \times 7)+7, y, 18, (x \times y)+x+x\} \\ &= \{3, 5x+5, 5x+7, 7x+5, 7x+7, y, 18, xy+2x\}. \end{aligned}$$

Note that the terms of G_σ are essentially polynomials in $x_1, \ldots, x_k$. Let g_σ denote the number of terms in G_σ. The polynomials in G_σ describe the points in the domain of $\mathcal{F}$ that may be accessed by the query $\psi(x_1, \ldots, x_k)$ (except terms which have nested $\mathcal{F}$'s; we will use m' to handle them later). The main idea will be to find a set of $k+1$ points that can not all be referred to in any instantiation of θ.

We view G_σ as a mapping of $\mathbb{N}^k$ to subsets of $\mathbb{N}$. of size $\leq g_\sigma$. For example, for the G_σ above, $G_\sigma(2,3) = \{3, 15, 17, 19, 18, 10\}$. Henceforth, we will denote G_σ by $G_\sigma(x_1, \ldots, x_k)$.

We construct the extension τ to σ as follows. We claim that there exists points, $p_1, \ldots, p_{k+1}$, not in the domain of σ such that $(\forall x_1) \cdots (\forall x_k)$

$$\{p_1, \ldots, p_{k+1}\} \not\subseteq G_\sigma(x_1, \ldots, x_k).$$

In other words, for all values x_1, ..., x_k, the points $p_1, \ldots, p_{k+1}$ cannot be simultaneously referred to in the query $\psi(x_1, \ldots, x_k)$.

To see that this is true let n be chosen such that $\binom{g_\sigma}{k+1} < n$. and consider the effect of G_σ (as a mapping) modulo n. Formally define G_σ^n to be the function that on input $(x_1, \ldots, x_k) \in \{0, 1, \ldots, n-1\}^k$ produces the multiset (repeated elements are allowed) obtained by taking every element in G_σ^n and reducing them mod n. (For example, for the above G_σ, $G_\sigma^5(2,3) = \{3, 0, 2, 4, 3, 0\}$.) For every $(x_1, \ldots, x_k) \in \{0, 1, \ldots, n-1\}^k$, $|G_\sigma^n(x_1, \ldots, x_k)| \leq g_\sigma$; hence the number of $k+1$ element subsets of $G_\sigma^n(x_1, \ldots, x_k)$ is $\leq \binom{g}{k+1}$. (By convention, a subset of a multiset may also be a multiset. For example, for the G_σ above, $\{3, 3, 0\}$ is a subset of $G_\sigma^5(2,3)$.) Since there are n^k elements in the domain of G_σ^n the number of $k+1$ element multisets $\{r_1, \ldots, r_{k+1}\}$ that can be a subset of some

$G^n_\sigma(x_1, \ldots, x_k)$ is $\leq n^k \cdot \binom{g}{k+1} < n^k \cdot n = n^{k+1}$. Since the number of possible $k+1$ element multisets that are subsets of $\{0, 1, \ldots, n-1\}$ is n^{k+1}, there exists a multiset $\{r_1, \ldots, r_{k+1}\}$ that is not a subset of any $G^n_\sigma(x_1, \ldots, x_k)$. Thus, for any set $\{p_1, \ldots, p_{k+1}\}$ such that for all $i \leq k+1$, $p_i \equiv r_i \pmod{n}$,

$$(\forall x_1, \ldots, x_k \in \mathbb{N})\{p_1, \ldots, p_{k+1}\} \not\subseteq G_\sigma(x_1, \ldots, x_k).$$

We use $p_1, \ldots, p_{k+1}$ in our construction of τ. Let m' be larger than $\max_i\{p_i\}$. Let τ be such that the positions $p_1, \ldots, p_{k+1}$ all have the same value $\geq m'$, and all other values are unique and $\geq m'$. To see that this will work, assume by way of contradiction that there is some extension f of $\sigma \cup \tau$ such that f makes θ true. In this case, there must then be values $a_1, \ldots, a_k$ such that f makes $\psi(a_1, \ldots, a_k)$ true. But we know that $\{p_1, \ldots, p_{k+1}\} \not\subseteq G_0(a_1, \ldots, a_k)$, so there must be some position $p_i \notin G_\sigma(a_1, \ldots, a_k)$. We claim that p_i is not referenced as an argument to $\mathcal{F}$ in the statement $\psi(a_1, \ldots, a_k)$. If it were, then it must be equal in value to some term of G_0, say t. If t does not involve $\mathcal{F}$, then $t \in G_\sigma$, a contradiction. If t does involve $\mathcal{F}$, then those occurrences of $\mathcal{F}$ must either take on values in the range of σ or values $\geq m'$. If any value $\geq m'$ is used in a place not eventually multiplied by zero, then the value of t will be $\geq m'$, contradicting $p_i < m'$. If all values not eventually multiplied by zero are in the range of σ, then $t \in G_\sigma$, a contradiction. Let f' be f with the position of p_i modified to contain a unique value. (If necessary, all unreferenced positions can be modified to permit this). Then f' will also make $\psi(a_1, \ldots, a_k)$ true, since $\psi(a_1, \ldots, a_k)$ does not reference the positions which have been changed. Thus, f' makes θ true, in contradiction to the assumption in the statement of this lemma. ⊠

We note that Lemma 18 can easily be extended to the language $[+, \times, <]$ (and therefore $[+, <]$) as well. In fact, Lemma 18 can be extended to any language of the form $[+, \times, P_1, P_2, \ldots, P_s]$, where $P_1, P_2, \ldots, P_s$ are any predicates whatsoever. This is because the term structure of the language still consists of multi-variate polynomials. In other words, the additional predicates do not affect which positions of $\mathcal{F}$ can be examined at the same time. In addition, Lemma 18 can be extended to any language where all the function symbols $\oplus(x_1, \ldots, x_m)$ have the property that, for all n,

$$(\oplus(x_1, \ldots, x_m) \bmod n) = (\oplus(x_1 \bmod n, \ldots, x_m \pmod n)) \bmod n).$$

We will not need this general version of Lemma 18 for this paper, but it may be useful in subsequent work.

THEOREM 19. For any $k > 0$, we have $Q_1^{k+1}EX_0[\ ] - \bigcup_{c=0}^{\infty} Q_1^k EX_c[+,<] \neq \emptyset$.

Proof:

For a function f, let

$$R(f,k) = \{(x_0, x_1, \ldots, x_k) \,|\, \bigwedge_{i \neq j} x_i \neq x_j \wedge\ f(x_0) = f(x_1) = \cdots = f(x_k)\}.$$

Let

$$T = \{f \,|\, \varphi_{f(0)} =^1 f, \text{ and } R(f,k) \text{ is finite, and}$$

$$\varphi_{f(0)}(z) = f(z) \text{ for all } x \geq \max\{\min y \in U \,|\, U \in R(f,k)\}.$$

We show that $T \in Q_1^{k+1}EX_0[\ ]$, by means of a QIM M as follows. Suppose $f \in T$. M asks the questions:

$$(\exists x_0)(\exists x_1)\cdots(\exists x_k)\ \left[(\bigwedge_i x_i \neq 0) \wedge (\bigwedge_{i\neq j} x_i \neq x_j)\mathcal{F}(x_0) = \mathcal{F}(x_1) = \cdots = \mathcal{F}(x_k)\right],$$

$$(\exists x_0)(\exists x_1)\cdots(\exists x_k)\ \left[(\bigwedge_i x_i \neq 0) \wedge (\bigwedge_i x_i \neq 1) \wedge (\bigwedge_{i\neq j} x_i \neq x_j)\mathcal{F}(x_0) = \mathcal{F}(x_1) = \cdots = \mathcal{F}(x_k)\right],$$

$$\vdots$$

$$(\exists x_0)(\exists x_1)\cdots(\exists x_k)\ \left[(\bigwedge_i x_i \neq 0) \wedge \ldots \wedge (\bigwedge_i x_i \neq n) \wedge (\bigwedge_{i\neq j} x_i \neq x_j)\mathcal{F}(x_0) = \mathcal{F}(x_1) = \cdots = \mathcal{F}(x_k)\right]$$

until a response of NO is obtained. Then, M determines the actual values of $f(0), f(1), \ldots,$ $f(n)$, and outputs an appropriately patched version of the program $f(0)$ as its only guess.

For all $c \geq 0$, we show that $T \notin Q_1^k EX_c[+,<]$. Suppose that M is a QIM that asks questions using the query language $[+,<]$ restricted to sentences with only k existential quantifiers. We construct a recursive function $f \in T$ that M does not infer within c mind changes. The construction proceeds in effective stages of finite extention. The initial segment of f constructed prior to stage s is denoted f^s. By implicit use of the recursion theorem, initialize $f^0 = \{(0, e)\}$. At each stage, we will have M's current query, which will be denoted (ambiguously) by ψ. In addition, we will have θ, which will be the disjunction of previous queries to which we have responded NO. (We keep θ to insure that the future construction remains consistent with the previous responses.) The value of this θ, on entry into stage s, will be denoted by θ^s.

As was the case in the proof of Theorem 4 the extensions considered during stage s below will cause M to ask questions which must be answered. The answering of the questions will necessitate fixing portions of the extension. The variable σ will be used to denote the (partial) extension to f^s currently determined. It will turn out that $f^s \subseteq \sigma \subseteq f^{s+1}$. Similarly, let $\vec{b}$ denote the answers to the questions that have been answered.

Begin stage s. Let m be the largest value in the range of f^s. Set $\theta = \theta^s$. Apply Lemma 18 to m and finite initial segment f^s to get an extension τ of f^s and $y \geq m$ occurring $k+1$ times in the range of τ. Choose x least such that $f(x) = y$. Let $\sigma = f^s \cup \tau - \{(x, y)\}$. Next, we perform the following two substages in parallel, until one terminates. If neither substage terminates, then f will be defined everywhere except at x.

Substage 1. Let $j = g(M(\vec{b}))$. Run $\varphi_j(x)$. If it converges before substage (2) is complete, then choose z least such that $z \geq m$, $x \notin$ range σ and $z \neq \varphi_j(x)$ and set

$$\sigma = \sigma \cup \{(x, z)\}$$

and execute the query answer procedure.

Query Answer Procedure. Let ψ be M's current query. There are two cases to consider. We can determine which of these cases holds by using a slight modification of the technique from Lemma 3 in to deal with the restriction on using values at most k times.

Case 1. If there is a finite extension $\tau \supset \sigma$ such that the range of τ contains only values $\geq m$ and no value appears in the range of τ more than k times and $\sigma \cup \tau$ makes ψ true, then set $\sigma = \sigma \cup \tau$ and answer the query YES.

Case 2. Otherwise, let z be the least value $> x$ not in the domain of σ, set $\sigma = \sigma \cup \{(z, m)\}$ (an arbitrary extension), set $\theta = \theta \vee \psi$ and answer the query NO.

End Query Answer Procedure.

Set $f^{s+1} = \sigma$, $\theta^{s+1} = \theta^s$ and go to stage $s+1$.

Substage 2. Answer the current query using the query answer procedure above. Obtain future queries from M and continue to answer them in this manner. If we encounter a mind change during this process, before substage (1) terminates then one of two cases applies. Suppose this mind change is the d^{th} one.

Case 1. $d < c+1$. Set $f^{s+1} = \sigma$, $\theta^{s+1} = \theta$ and go to stage $s+1$.

Case 2. $d = c+1$. Define f:

$$f(z) = \begin{cases} \sigma(z) & \text{if } z \in \text{domain } \sigma, \\ \text{undefined} & \text{if } z = x, \\ z + m & \text{otherwise.} \end{cases}$$

and terminate stage s and do not execute any stages $s' > s$.

End stage s.

If all stages of the construction are executed, then either M makes $c+1$ mind changes while inferring f, or M's final guess is wrong infinitely often. If the construction stalls at some stage, then f is undefined at some point x. However, the function $f' = f \cup \{(x, y)\}$ is in T, and M's final guess on inferring f' is not a total function. ☒

COROLLARY 20. For any $k > 0$, we have $Q_1^{k+1}EX_0[<] - \bigcup_{c=0}^{\infty} Q_1^k EX_c[+,<] \neq \emptyset$.

COROLLARY 21. For all $c \in \mathbb{N}$

1. $Q_1^k EX_c[S,<] \subset Q_1^{k+1}EX_c[S,<]$.
2. $Q_1^k EX_c[+,<] \subset Q_1^{k+1}EX_c[+,<]$.

THEOREM 22. For any reasonable language L, for any $k > 0$ and any $c \geq 0$, we have $Q_1^k EX_{c+1}[L] - Q_1^k EX_c[L] \neq \emptyset$.

Proof: In [12] it was shown that $EX_{c+1} - Q_1 EX_c[L]$ is nonempty for any reasonable language L. The result follows. ⊠

Finally, in light of the definitions of this section, a careful examination of the proof of Theorem 8 reveils a proof of a stronger result. We conclude with a statment of this result.

THEOREM 23. $Q_1^1 EX_c[\,] - [1,c]BC \neq \emptyset$.

VII. Conclusions

The results of [13] have been extended in several directions. Inference by asking questions has been related to team learning. The quantifier structure of queries has been used as a measure of articulation. Not surprisingly, more articulate query machines are more capable learners. What is perhaps surprising are the preliminary results indicating that the number of quantifiers may be important. Our most general results are graphically summerized below.

SUMMARY

By Corollary 11 and results from [13]

$$\begin{array}{ccccccc}
EX & \subset & [1,2]EX & \subset & [1,3]EX & \subset & \dots \\
\cup & & \cup & & \cup & & \\
Q_1EX_0[S,<] & \subset & Q_1EX_1[S,<] & \subset & Q_1EX_2[S,<] & \subset & \dots
\end{array}$$

By Corollary 11 and results from [12,26]

$$\begin{array}{ccccccc}
EX & \subset & [1,2]EX & \subset & [1,3]EX & \subset & \dots \\
\cup & & \cup & & \cup & & \\
Q_1EX_0[+,<] & \subset & Q_1EX_1[+,<] & \subset & Q_1EX_2[+,<] & \subset & \dots
\end{array}$$

By Theorem 2, Corollary 11, and results from [12,26]

$$\begin{array}{ccccccc}
EX & \subset & [1,2]EX & \subset & [1,3]EX & \subset & \dots \\
\cup & & \cup & & \cup & & \\
Q_1EX_0[+,\times] & \subset & Q_1EX_1[+,\times] & \subset & Q_1EX_2[+,\times] & \subset & \dots
\end{array}$$

All inclusions not shown in the above diagrams represent known incomparabilites, except for the diagram concerning the query language $[+,\times]$. Recall that $[1,n]EX$ is precisely the same collection of sets of functions that can be *probabilistically* inferred by an inference machine with probability $1/n$ [18].

VIII. Acknowledgements

Computer time was provided by the Department of Computer Science at the University of Maryland. The National Science Foundation supported two of the authors. Peter Montgomery supplied a crucial trick that was used in the proof of Lemma 18.

References

1. Angluin, D. Learning k-term DNF formulas using queries and counter-examples. Department of Computer Science TR-559, Yale University, New Haven, CT, 1987.
2. Angluin, D. Learning k bounded context-free grammars. Department of Computer Science TR-557, Yale University, New Haven, CT, 1987.
3. Angluin, D. Queries and concept learning. *Machine Learning 2* (1988), 319–342.

4. ANGLUIN, D. Equivalence queries and approximate fingerprints. In *Proceedings of the 1989 Workshop on Computational Learning Theory*, M. Warmuth, Ed., Morgan Kaufmann, San Mateo, CA., 1989.

5. ANGLUIN, D. AND SMITH, C. H. Inductive inference: theory and methods. *Computing Surveys 15* (1983), 237–269.

6. ANGLUIN, D. AND SMITH, C. H. Inductive inference. In *Encyclopedia of Artificial Intelligence*, S. Shapiro, Ed., John Wiley and Sons Inc., 1987.

7. BERMAN, P. AND ROOS, R. Learning one–counter languages in polynomial time. 28^{th} *Annual FOCS conference* (1987), 61–67.

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

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

10. CASE, J. AND SMITH, C. Comparison of identification criteria for machine inductive inference. *Theoretical Computer Science 25,* 2 (1983), 193–220.

11. FREIVALDS, R., SMITH, C., AND VELAUTHAPILLAI, M. Trade–offs amongst parameters effecting the inductive inferribility of classes of recursive functions. *Information and Computation 82,* 3 (1989), 323–349.

12. GASARCH, W., PLESZKOCH, M., AND SOLOVAY, R. *Learning via queries with plus and times.* manuscript.

13. GASARCH, W. AND SMITH, C. Learning via Queries. *Journal of the ACM* (1991). acceptance indicated.

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

15. HELLERSTEIN, L. AND KARPINSKI, M. Learning read-once formulas using membership queries. In *Proceedings of the 1989 Workshop on Computational Learning Theory*, M. Warmuth, Ed., Morgan Kaufmann, San Mateo, CA., 1989.

16. ISHIZAKA, H. Learning simple deterministic languages. In *Proceedings of the 1989 Workshop on Computational Learning Theory*, M. Warmuth, Ed., Morgan Kaufmann, San Mateo, CA., 1989.

17. MACHTEY, M. AND YOUNG, P. *An Introduction to the General Theory of Algorithms.* North-Holland, New York, New York, 1978.

18. PITT, L. A Characterization of Probabilistic Inference. *Journal of the ACM 36,* 2 (1989), 383–433.

19. PITT, L. AND SMITH, C. Probability and plurality for aggregations of learning machines. *Information and Computation 77* (1988), 77–92.

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

21. ROYER, J. S. Inductive inference of approximations. *Information and Control 70,* 2/3 (1986), 156–178.

22. Sakakibara, Y. *Inferring parsers of context-free languages from structural examples.* Fujitsu International Institute for Advanced Study of Social Information Science, Numazu, Japan, 1981.

23. Sakakibara, Y. Inductive inference of logic programs based on algebraic semantics. Technical Report 79, Fujitsu International Institute for Advanced Study of Social Information Science, Numazu, Japan, 1987.

24. Shapiro, E. *Algorithmic programming debugging.* MIT Press, Cambridge, MA, 1983.

25. Smith, C. A note on arbitrarily complex recursive functions. *Notre Dame Journal of Formal Logic 29,* 2 (1988), 198–207.

26. Smith, C. H. The power of pluralism for automatic program synthesis. *Journal of the ACM 29,* 4 (1982), 1144–1165.

27. Smith, C. H. and Velauthapillai, M. On the inference of approximate explanations. *Theoretical Computer Science.* To appear.

28. Valiant, L. G. A theory of the learnable. *Communications of the ACM 27,* 11 (1984), 1134–1142.

29. Wiehagen, R. Limes-erkennung rekursiver funktionen durch spezielle strategien. *Elektronische Informationsverarbeitung und Kybernetik 12* (1976), 93–99.