Learning via Queries[†]

by

William I. Gasarch
and
Carl H. Smith[‡]
Department of Computer Science and
Institute for Advanced Computer Studies
The University of Maryland
College Park, MD 20742

I. Introduction

Computer scientists have become interested in inductive inference as a form of machine learning, in part because of artificial intelligence considerations, see [7,8] and the references therein. The contribution of this paper is a formal verification that a technique that humans use to learn (asking questions) is also advantageous to machines that are trying to learn. Inquisitiveness is the driving force behind all of science. Formal learning situations almost always provide opportunities for the learner(s) to ask questions. The notion of learning by asking questions is formalized, and results are obtained that show asking questions to be a powerful learning technique. As will be seen, our results may be of relevance to human learning, even though, for many of our results, the trainer supplying the answers to the queries must surpass the algorithmic in ability.

[†] Supported by NSF grants CCR 8701104 and CCR 8803641. A preliminary version of this paper was presented at the 29th Symposium on the Foundations of Computing in White Plains, NY, October 1988.

[‡] Some of the research reported on below was done while the the second author was on leave at The National Science Foundation. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Traditional work in the field [7,8] and the more recent work on Valiant's [43] model of time bounded learning [25,26,31] regard the inference machines as *passive* recipients of data regarded as examples of the behavior of the function to be learned. Subsequently, Angluin devised an elegant formulation of a teacher and learner paradigm [3]. In this work the inference device is allowed, in addition to receiving data, to ask a teacher questions. These questions, moreover, may be of a more general kind than merely "is $x \in L$?" They can also be of the kind "is this a good model of L and if not, why not?" Besides asking questions regarding membership and equivalence with another language, Angluin permits questions of subset and superset [5]. The basic paradigm of asking questions has been applied to DNF formulas [4], context-free grammars [2], deterministic one-counter automata [9], deterministic bottom up tree automata [38], deterministic skeletal automata [37], Prolog programs [39]. Valiant also considered the issue briefly. [43]. For a nice summary of some of these results see [5,6].

The focus of this paper is a recursion theoretic version of Angluin's teacher and learner paradigm. We compare learning by asking questions to learning by passively reading data. Although some of our results hold for many query languages, generally the extent of what can be learned by queries is largely dependent on the particular language which the inference mechanism uses to phrase its questions to the teacher. Formulating questions may not consume more of the learner's computational resources as the query language becomes more expressive, but in many cases, the teacher must be more computationally gifted than the learner in order to properly answer the questions. Our own pedagogical experiences indicate that sometimes students ask questions that their teachers cannot answer. Analogously, we allow the learning devices considered below to ask undecidable questions. This means that the "teachers" we employ in what follows cannot always be emulated by a mechanistic device. Our results are summarized in the remainder of this section.

Consider a reasonable query language in which all of the function and predicate symbols stand for recursive entities. We prove that the learning potential of machines that ask

only quantifier-free questions is precisely the same as the learning potential of standard, passive, inductive inference machines; but being able to ask existential questions enhances the learning potential. When the inference devices are constrained to output a single conjecture and ask only existential questions, they do not exceed the power of passive inference devices. However, if more than one conjecture is allowed, then it is possible to learn phenomena that are not learnable by passive inductive inference devices.

Using a result of Davis, Putnam, Robinson and Matijasevič [19,29], we prove that inference machines that are allowed to ask existential questions with plus and times can learn all the recursive functions. With this in mind, we concentrate on inference machines that ask questions formulated in weaker languages.

If the language consists of plus and less than and only existential queries are allowed, then the set of all recursive functions cannot be learned. Since this language is reasonable, machines constrained to ask existential questions and only make one conjecture do not exceed the power of passive inference devices. However, if queries with an alternation of quantifiers are allowed, then it is possible to learn phenomena that are not learnable passively. This indicates some sort of trade off between alternations of quantifications and the number of times an inference device may change its conjecture.

If the language consists of successor and less than, then the set of recursive functions cannot be learned. For this language the classes that can be learned with only one conjecture is a proper subset of what can be learned passively. The theory of ω-automata [11,16,30,33,36] is employed in the proof of these results to restrict, without precisely specifying, the future of some set under construction.

II. Technical Summary

In this section we formalize our notions and state our results precisely. Our machines will learn recursive functions which are taken to encode an arbitrary phenomenon, see the discussion in [14]. 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 [10,24] (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. This is denoted by $M(f)\downarrow$. Learning has occurred since we can use the program output by the IIM to predict the value of $f(x)$, for all x, after having seen only finitely many examples. 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 σ is an initial segment of some function f, then $M(\sigma)$ denotes the last conjecture about f made by M, after seeing all of σ as input, but prior to requesting additional input.

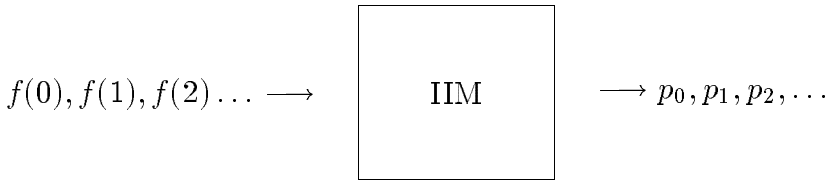

Figure 1

Each time an IIM outputs a conjecture we say that another *learning trial* has been completed. Since it is never known if enough trials have been completed, it is sometimes desirable to fix an a priori bound on the number of trials that will be permitted. 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. The number of mind changes needed by an IIM to make a successful inference is a crude measure of the complexity of performing the inference [17].

A *query inference machine* (QIM) is an algorithmic device that asks a teacher questions about some unknown function, and while doing so, outputs programs. In this way, the QIM learns about some phenomenon, encoded by a recursive function, by asking finitely many questions. We assume that the teacher always returns the correct answer to any

question asked by a QIM. The questions are formulated in some query language L. A variety of different query languages are considered. The languages that we consider have different expressive power. The more expressive the query language, the more questions the QIM can ask. In a sense, giving a QIM a more expressive query language to use makes the QIM more articulate.

The results we obtain classify the learning potential of QIMs with respect to their abilities to articulate appropriate questions. Some of the query languages discussed below are very powerful indeed. The teacher who answers the QIM's questions must have profound knowledge of the function f to be inferred. Merely having an oracle for f will not suffice, e.g. consider the query "$\exists x[f(x) = 0]$?" The teacher may need an oracle for the halting problem. Hence, the teacher's "algorithm" to answer questions most likely will be of a higher degree of unsolvability than the recursive QIM. Exactly what the teacher needs may depend on both the function f and the query language L. In some cases the question may be effectively undecidable. Note that in logic programming, queries involving negation and a function symbol (as is the case with our query languages) posed in the presence of a closed world assumption are also undecidable [40]. The issue of how much (and what type of) computational resources are needed to answer the questions posed by a QIM are not investigated in this work.

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 guess which is a (possibly null) program e, and a question ψ, see Figure 2. Define two functions g (*guess*) and q (*query*) such that if $M(\vec{b}) = (e, \psi)$ then $g(M(\vec{b})) = e$ and $q(M(\vec{b})) = \psi$. By convention, all questions are assumed to be sentences in prenex normal form (quantifiers followed by a quantifier-free formula, called the matrix of the formula) and questions containing quantifiers are assumed to begin with an existential quantifier. This convention entails no loss of generality and serves to reduce the number of cases needed in several proofs. 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 $QEX[L](M)$ as M varies over all QIMs that use the query language L. $QEX_c[L]$ is defined similarly.

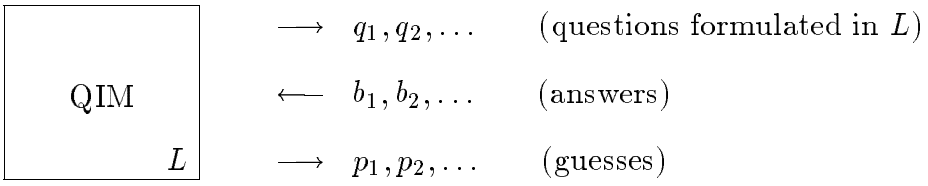

Figure 2

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 language. Of concern to us is the alternations between blocks of existential and universal 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 $a > 0$ alternations between blocks of universal and existential quantifiers, then we say that $f \in Q_{a+1}EX[L](M)$. The classes $Q_aEX[L]$ and $Q_aEX_c[L]$ are defined analogously. By convention, if a QIM restricted to c mind changes actually achieves that bound, then it will ask no further questions.

Now we introduce the query 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 unary 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 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 $[+, <]$ and is Presburger arithmetic with a function symbol. The language that has auxiliary symbols for plus and times is denoted by $[+, \times]$ and is Peano

arithmetic with a function symbol. The language with extra symbols for successor and less than is denoted by $[S, <]$, where S indicates the symbol for the successor operation. This last language is based on $S1S$ (second order theory of one successor) [11,33]. Presburger arithmetic, Peano arithmetic, and $S1S$, have been studied extensively, indicating their naturalness. Furthermore, these languages display a spectrum of decidability properties: Peano arithmetic is undecidable, Presburger arithmetic is decidable but its second order theory is undecidable, and $S1S$ is a decidable second order theory.

We denote languages where quantification over sets is allowed by appending "2^{nd}" to the list of operators, even though it is not an operator. 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*.

Some examples of sentences in $[<]$ without quantifiers are:

$$(\mathcal{F}(7) = 13) \wedge \neg(\mathcal{F}(3) < 10)$$

$$\mathcal{F}(\mathcal{F}(3)) = 74.$$

Some examples of sentences in $[+, <]$ with a single type of quantifier are:

$$\exists x \exists y [\mathcal{F}(x) < y \wedge \mathcal{F}(y) = x + y]$$

$$\forall x \forall y [\mathcal{F}(x) = y \Leftrightarrow \mathcal{F}(y) = x].$$

Sentences in $[+, \times]$ with no alternations of quantifiers include:

$$\exists x \exists y \exists z [\mathcal{F}(x) = \mathcal{F}(y) \wedge x = y \times z]$$

$$\exists x \exists y \exists z [(x \times x) + (y \times y \times y) = 3 \times z \times z \times z].$$

An example of a sentence in $[S]$, written informally, is:

$$\exists y [y > 100 \wedge \mathcal{F}(y) = \mathcal{F}(y+1) = \mathcal{F}(y+2) = \mathcal{F}(y+3) = \cdots = \mathcal{F}(y+17) = 1].$$

An example of a sentence with an alternation of quantifiers is:

$$\forall x \exists y [y > x \wedge \mathcal{F}(y) = 0].$$

We conclude this section with an enumeration of our major results. Henceforth, $\subseteq$ denotes subset and $\subset$ denotes *proper* subset. The first result shows that, for any reasonable query language, asking quantifier-free questions results in learning procedures that are equivalent in power to standard, passive inference machines.

1. $Q_0 EX[\star] = EX$.

However, if the QIM is not allowed to change its conjecture, then allowing a quantifier does not yield a learning paradigm more powerful than traditional inference.

2. $Q_1 EX_0[\star] \subseteq EX$.

On the other hand, allowing a single mind change and a single type of quantifier does allow one to infer sets not in EX. Hence, quantification enhances considerably the ability of a QIM to articulate queries.

3. $Q_1 EX_1[\star] - EX \neq \emptyset$.

The next result concerns inference by queries formulated in the language $[+, \times]$. Imbuing QIMs with this query language with quantifiers essentially gives them the ability to ask undecidable questions. This is not only impractical but, as the next result indicates, yields an inference paradigm that is not interesting theoretically.

4. $Q_1 EX[+, \times] = Q_2 EX_0[+, \times] = Q_2 EX[+, \times] = Q_3 EX[+, \times] \ldots$, and each contains the set of all recursive functions.

The above result shows how important it is to choose an appropriate query language.

The next group of results concerns Presburger arithmetic. A most fundamental result that, to a large extent, motivates this paper is that the use of quantifiers allows asking more powerful questions which, in turn, enables a larger class of functions to be learned.

5. $Q_0 EX[+, <] \subset Q_1 EX[+, <]$.

We believe that the above result establishes what will be the base of an infinite hierarchy defined in terms of alternations of blocks of quantifiers. Subsequent to our work it was discovered that $Q_1EX[+,<] \subset Q_2EX[+,<]$ [22].

6. $Q_1EX[+,<]$ does not contain the set of all recursive functions.

Subsequent to our work it was discovered that the set of recursive functions is not a member of $QEX[+,<]$ [23].

There seems to be a tradeoff between the number of mind changes and the number of alternations of quantifiers in the questions asked by the QIM. This trade off is evidenced by the next result.

7. $Q_2EX_0[+,<] - EX \neq \emptyset$.

The final section of this paper concerns query machines that ask questions using a language with a successor symbol. Again, we find that quantifiers enable the posing of more questions which are beneficial to the inference process. In contrast to the language $[+,<]$, here we obtain results about query languages that allow an unbounded number of alternations of quantifiers. The proofs of these results use a deep theorem about ω-automata and their relation to $S1S$ [11,33], and several other lemmas about ω-regular languages.

8. $Q_0EX[S,<] \subset Q_1EX[S,<]$.
9. $QEX_a[S,<]$ and EX are incomparable if $a \geq 1$. If only the inference of sets is considered, then $QEX_0 \subset EX$. This can be extended to the inference of functions with a known finite range.
10. $QEX[S,<]$ does not contain the set of all recursive sets.

This last result also holds if we allow various enhancements to the language $[S,<]$.

III. Definitions and technical lemmas

Throughout this paper, φ_0, φ_1, φ_2, ... denotes an acceptable programming system [27], also known as a Gödel numbering of the partial recursive functions [34]. The function φ_e is said to be computed by the program e. Let $\mathcal{R}$ denote the class of all recursive

functions. Throughout this section we assume that L is an arbitrary, but fixed, reasonable query language.

If f is a total function and ψ is a sentence (i.e. no free variables) in L then $f \models \psi$ can be defined in a natural way that corresponds to "if in ψ we interpret $\mathcal{F}$ as f then ψ is true." When a QIM is inferring a function f and makes the query ψ then the answer received is YES (NO) if $f \models \psi$ ($f \models \neg\psi$). Our definition of "$\models$" is similar to the definition commonly used in model theory [15].

Throughout this paper we will construct functions in effective stages of finite extension. At stage s of such a construction we will commonly have f_s (a finite function) and ψ (a sentence). At this point, we will be trying to find a finite extension f_{s+1} of f_s that forces ψ or $\neg\psi$ to be true. Hence we will define $g \models \psi$ for a finite function g and certain types of ψ. Several technical results about $\models$ are proven.

Recall that for our purposes the language L contains the usual logical operators, the symbol $\mathcal{F}$, relational symbols, and operational symbols. The relational and operational symbols are to be interpreted in the obvious way (e.g. "+" denotes plus and "<" denotes less than). It is assumed that any sentence without quantifiers or the symbol $\mathcal{F}$ can be effectively assigned a truth value (e.g., $2 + 2 < 5$ is TRUE). A further assumption is that if $a_1,\ a_2,\ \ldots,\ a_n \in \mathbb{N}$ and $\otimes$ represents an n-place operation in the language L, then $\otimes(a_1, a_2, \ldots, a_n) \in \mathbb{N}$ and can be found effectively.

The notion of *term* is now defined. Let L be any language. Then:

1. Any constant or variable is a *term*.
2. If $t_1,\ t_2,\ \ldots,\ t_n$ are terms and $\otimes$ represents an n-ary operation symbol in L, then $\otimes(t_1, \ldots, t_n)$ is a term.
3. If t is a term then $\mathcal{F}(t)$ is a term.

A term whose only variables are x_1, x_2, ..., x_n will be written $t(x_1, \ldots, x_n)$. Suppose $t(x_1, \ldots, x_n)$ is a term, a_1, ..., $a_n \in \mathbb{N}$, and f is a partial function. The constant $t(a_1, \ldots, a_n)[f]$ is defined (if at all) inductively as follows:

If t is a constant c, then $t[f] = c$.

If $t(x)$ is just the variable x, then $t(a)[f] = a$.

If $t = \mathcal{F}(s(x_1, \ldots, x_n))$, then

$$t(a_1, \ldots, a_n)[f] = \begin{cases} \text{undefined} & \text{if } s(a_1, \ldots, a_n) \text{ is undefined;} \\ \text{undefined} & \text{if } s(a_1, \ldots, a_n) \notin \text{domain of } f; \\ f(s(a_1, \ldots, a_n)) & \text{otherwise.} \end{cases}$$

If $t = \otimes(t_1(x_1, \ldots, x_{n_1}), \ldots, t_m(x_1, \ldots, x_{n_m}))$, then let

$$t(a_1, \ldots, a_n)[f] = \otimes(t_1(a_1, \ldots, a_{n_1})[f], \ldots, t_m(a_1, \ldots, a_{n_m})[f])$$

if all the subterms are defined, and undefined otherwise.

LEMMA 1. If $t(a_1, \ldots, a_n)[f]$ is defined then there exists a finite function $f' \subset f$ such that $t(a_1, \ldots, a_n)[f']$ is defined.

Proof: By induction on the structure of t. ☒

Let f be a total function and ψ be a sentence in L. We define $f \models \psi$ for quantifier free ψ and then extend to the quantifier case by induction. This defines $f \models \psi$, for all ψ, because we have assumed that ψ is in prenex normal form.

1. If ψ is quantifier free then we can write $\psi = \phi(t_1, \ldots, t_m)$, where ϕ contains only predicates and logical connectives, but no variables. Then, $f \models \psi$ iff $\phi(t_1[f], \ldots, t_m[f])$ is true.
2. If $\psi = \exists x_1, \ldots, x_n \theta(x_1, \ldots, x_n)$ then $f \models \psi$ iff there exist a_1, ..., $a_n \in \mathbb{N}$ such that $f \models \theta(a_1, \ldots, a_n)$.
3. If $\psi = \forall x_1, \ldots, x_n \theta(x_1, \ldots, x_n)$ then $f \models \psi$ iff for all a_1, ..., $a_n \in \mathbb{N}$, $f \models \theta(a_1, \ldots, a_n)$.

Next, we define $\langle f, p\rangle \models \psi$ for f a finite function, $p \in \mathbb{N}$ and, ψ an existential sentence or the negation of one, as follows:

1. If ψ is quantifier free then we can write $\psi = \phi(t_1, \ldots, t_m)$ where ϕ contains only predicates and logical connectives, but no variables. Then, $\langle f, p\rangle \models \psi$ iff every $t_i[f]$ is defined and $\phi(t_1[f], \ldots, t_m[f])$ is true. Note that since f is total, each $t_i[f]$ must be defined.
2. If $\psi = \exists x_1, \ldots, x_n \theta(x_1, \ldots, x_n)$ then $\langle f, p\rangle \models \psi$ iff there exists $a_1, \ldots, a_n \in \{0, 1, \ldots, p\}$ such that $f \models \theta(a_1, \ldots, a_n)$.
3. If $\psi = \neg\exists x_1, \ldots, x_n \theta(x_1, \ldots, x_n)$ then $\langle f, p\rangle \models \psi$ iff for every finite extension f' of f it is not the case that $\langle f', p\rangle \models \exists x_1, \ldots, x_n \theta(x_1, \ldots, x_n)$.

If f is a finite function and ψ is an existential sentence then $f \models \psi$ iff there exists a p such that $\langle f, p\rangle \models \psi$. If f is a finite function and ψ is the negation of an existential sentence then $f \models \psi$ iff for all p it is not the case that $\langle f, p\rangle \models \neg\psi$. If $f \models \psi$ then, even though f is finite, there is enough information in f to validate ψ.

The reason that $f \models \psi$ can be defined for f finite and ψ existential is that if an existential formula is true, there must be a finite witness to its truth. Since this is not the case for harder formulas, there is no clearly sensible definition of $f \models \psi$ for f finite and ψ of the form $(\exists x_1, \ldots, x_n)(\forall y_1, \ldots, y_m)\theta(x_1, \ldots, x_n, y_1, \ldots, y_m)$.

LEMMA 2. If f is total and ψ is an existential sentence then $f \models \psi$ iff there exists a finite function $f' \subset f$ such that $f' \models \psi$.

Proof: By induction on the structure of ψ using the definition of $f \models \psi$ and Lemma 1. ☒

LEMMA 3. Suppose L is such that the problem of deciding the truth of existential sentences formulated in L, without using the function symbol $\mathcal{F}$, is decidable. Then the set of all pairs (f, ψ) such that:

i. f is a finite function, and

ii. ψ is an existential sentence of L, and

iii. there exists a finite function $f' \supset f$ such that $f' \models \psi$

is decidable.

Proof: Input (f, ψ) where f is a finite function and ψ is an existential sentence of L. The following algorithm reduces the problem for sentences ψ with nested occurrences of $\mathcal{F}$ to the problem for sentences without nested occurrences of $\mathcal{F}$. Assume

$$\psi = \exists x_1, \ldots, x_n[\theta(x_1, \ldots, x_n)].$$

Repeat the following process iteratively until there are no nested occurrences of $\mathcal{F}$. For each term $\mathcal{F}(s)$ where $\mathcal{F}(s)$ is a subterm of a term t such that $\mathcal{F}(t)$ appears in ψ, replace ψ with a sentence obtained as follows:

a. replace all occurrences of $\mathcal{F}(s)$ in the matrix of ψ with a new variable z,

b. insert the conjunct "$\mathcal{F}(s) = z$" in the matrix of ψ, and

c. add a "$\exists z$" to the front of ψ.

d. if the term $\mathcal{F}(a)$ appears, where a is in the domain of f, then replace "$\mathcal{F}(a)$" with the value of $f(a)$.

After the process is complete, let $\mathcal{F}(t_1)$, ..., $\mathcal{F}(t_m)$ be a list of the occurrences of $\mathcal{F}$ in ψ. Note that the terms t_1, ..., t_m do not contain $\mathcal{F}$. The final version of ψ is

$$\psi = \exists z_1, \ldots, z_r, x_1, \ldots, x_n[\theta'(z_1, \ldots, z_r, x_1, \ldots, x_n, \mathcal{F}(t_1), \ldots, \mathcal{F}(t_m))].$$

Let Ψ be

$$\exists z_1, \ldots, z_r, x_1, \ldots, x_n, w_1, \ldots, w_m \Big[\theta'(z_1, \ldots, z_r, x_1, \ldots, x_n, w_1, \ldots, w_m) \wedge \bigwedge_{i=1}^{m} \bigwedge_{j \in \mathrm{domain}(f)} t_i = j \Rightarrow w_i = f(j) \wedge \bigwedge_{1 \leq i,j \leq m} t_i = t_j \Rightarrow w_i = w_j \Big].$$

Note that the sentence Ψ has no occurrence of $\mathcal{F}$ in it and so it can be decided. It is easy to show that Ψ is true iff $\langle f, \psi \rangle$ satisfies conditions *i*, *ii* and *iii*. ☒

IV. Comparison with passive inference

In this section we compare EX to classes of the form $Q_i EX_j[\star]$. Recall that the "$\star$" indicates that these results hold for any reasonable query language. Comparison of EX to inference with specific query languages appears in subsequent sections. Recall that $M(\vec{b})$ is an ordered pair of a guess, $g(\vec{b})$ (a program), and question, $q(\vec{b})$.

PROPOSITION 4. $Q_0 EX[\star] = EX$.

Proof: Let S be in $Q_0 EX[\star]$ as witnessed by the QIM M. Note that M can only pose queries involving constants and $\mathcal{F}$ applied to constants. An IIM can simulate M by waiting for enough input to answer the query. With sufficient data, the IIM can deduce the answers to such queries.

On the other hand, suppose that $S \subseteq EX(M)$, where M is an IIM. Let $f \in S$. A QIM can simulate M on input from f by asking "$\mathcal{F}(0) = 0$?" "$\mathcal{F}(0) = 1$?" until the value of $f(0)$ is known. The QIM then outputs $M((0, f(0)))$. In a similar fashion, the QIM can discover $f(1)$, $f(2)$, ... and simulate M on larger and larger initial segments. ☒

THEOREM 5. $Q_1EX_0[\star] \subseteq EX$.

Proof: Suppose M is a QIM and $f \in Q_1EX_0[\star](M)$. Let f_s be the restriction of f to domain $\{0, \ldots, s\}$. Since M, when inferring f, can only output a single conjecture, there is a sequence of answers $\vec{b} \in \{0,1\}^\star$ such that the i^{th} bit of $\vec{b}$ correctly answers the i^{th} query and $g(M(\vec{b}))$ is defined and is a correct program for f. Once the correct $\vec{b}$ is found, the IIM will output a correct guess for f. A key point is that, by Lemma 2, if ψ is an existential query, then $f \models \psi$ iff $\exists s(\langle f_s, s\rangle \models \psi)$.

We now describe an algorithm for an IIM that infers f. During each stage of the construction a vector $\vec{b}$, which approximates the vector of true answers, is constructed and used. We later show that, in the limit, $\vec{b}$ is correct.

Stage s. Set $\vec{b} = \lambda$, the empty string.

For $i := 1$ to s, do the following

$\psi := q(M(\vec{b}))$

If $\langle f_s, s\rangle \models \psi$ then $\vec{b} := \vec{b}1$ else $\vec{b} := \vec{b}0$

Output $g(M(\vec{b}))$.

End stage s.

Let s be so large that

(i) the number of distinct queries made by the QIM M while trying to infer f is $s_0 \leq s$, and

(ii) if the queries made by M while trying to infer f are ψ_1, ψ_2, ..., ψ_{s_0} then for all $i \leq s_0$ such that ψ_i is true, $\langle f_s, s\rangle \models \psi_i$.

By Lemma 2, such an s exists. It is easy to see that for all $t \geq s$ the vector of answers $\vec{b}$ produced in stage t is correct. Hence, the output produced at stage t is the index for f that M produced. This index was assumed to be correct. ☒

The following theorem shows that Theorem 5 cannot be improved to one mind change. The proof uses a set S which is a baroque variation of a set used in [14].

THEOREM 6. $Q_1EX_1[\star] - EX \neq \emptyset$.

Proof: We define a set $S \in Q_1EX_1[\star] - EX$. Let S be the set of all recursive functions f such that there exists an $e \in \mathbb{N}$ such that

1. for infinitely many x, $f(x) = e$, and
2. if $f(x) = a \neq e$, then x is the unique number that maps to a, and
3. either

   $\varphi_e = f$, or

   $f(x) = e$ for all but finitely many x.

A QIM witnessing that $S \in Q_1EX_1[\star]$ is defined as follows. First find e by asking:

$$\exists x, y[x \neq y \wedge \mathcal{F}(x) = \mathcal{F}(y) = 0]$$

$$\exists x, y[x \neq y \wedge \mathcal{F}(x) = \mathcal{F}(y) = 1]$$

$$\vdots$$

$$\exists x, y[x \neq y \wedge \mathcal{F}(x) = \mathcal{F}(y) = e]$$

until a YES answer is encountered. Output that value of e as the first guess. Then determine if the range of f is finite by asking:

$$\exists x[\mathcal{F}(x) \neq e]$$

$$\exists x[x \neq 0 \wedge \mathcal{F}(x) \neq e]$$

$$\vdots$$

$$\exists x[x \neq 0 \wedge x \neq 1 \wedge \cdots \wedge x \neq b \wedge \mathcal{F}(x) \neq e]$$

until (if ever) a NO answer is encountered. If a NO answer is never encountered, then e, the QIM's first conjecture, is correct. If a NO answer is encountered, then, using the techniques of Proposition 4, find the values of $f(0)$, $f(1)$, ..., $f(b)$ and output a program that matches those values and has value e thereafter.

Now we show that $S \notin EX$. Suppose M is an IIM. We construct an $f \in S - EX(M)$. A function φ_e is constructed in effective stages of finite extension. Either φ_e is the desired

f or φ_e is a finite function with an extension that will serve as the desired f. The finite initial segment of φ_e determined prior to stage s is denoted by φ_e^s. By implicit use of the recursion theorem, we know the index e of the function we are about to construct.

Initialize $\varphi_e^0 = \{(0, e)\}$. Execute the following stages in their natural order.

Begin stage s. For convenience, let $\sigma = \varphi_e^s$. Let x be the least number not in the domain of σ. Simultaneously execute the following two substages.

Substage 1. Try to diagonalize against the current guess. Simulate the computation of $\varphi_{M(\sigma)}(x)$. If it converges before a mind change is found in the next substage, choose the least $y \notin (\{\sigma(z) \mid z < x\} \cup \{\varphi_{M(\sigma)}(x)\})$ and set

$$\varphi_e^{s+1} = \varphi_e^s \cup \{(x, y), (x+1, e)\},$$

and go to stage $s + 1$.

Substage 2. Try to force M to output a new conjecture. Look for a $\tau \supset \sigma$ such that

the domain of τ is an initial segment of $\mathbb{N}$, and

$\forall z \geq x$, if $\tau(z)$ is defined, then $\tau(z) = e$, and

$M(\tau) \neq M(\sigma)$.

If such a τ is found before $\varphi_{M(\sigma)}(x)$ converges in the first substage, then set $\varphi_e^{s+1} = \tau$ and go to stage $s + 1$

End stage s.

If every stage s terminates, then φ_e is a recursive function. If not, then φ_e is a finite function whose domain is an initial segment of the natural numbers. The two cases are considered separately.

Case 1. φ_e is a recursive function. Let $f = \varphi_e$. If the first substage extends φ_e at stage s then this extension defines $f(x) = y \neq e$ and $f(x+1) = e$ for some values of x and y unique to stage s. If the second substage extends φ_e at stage s then only e's are added to the range of φ_e at stage s. Hence, $f \in S$. Suppose that $M(f) \downarrow= M(\varphi_e^s) = i$, for some s, as otherwise M cannot possibly infer f. Then at every stage $t \geq s$, φ_e is extended by the first substage. Each such extension defines $f(x) \neq \varphi_i(x)$ for some x, which is a contradiction. Hence, $f \notin EX(M)$.

Case 2. For some s, $\varphi_e = \varphi_e^s$. Let x be the least number not in the domain of φ_e^s. Let f be defined by:

$$f(y) = \begin{cases} \varphi_e(y) & \text{if } y < x, \\ e & \text{if } y \geq x \end{cases}$$

It is easy to see that $f \in S$. By the failure of the second substage to extend φ_e^s, $M(f) \downarrow= M(\varphi_e^s) = j$. By the failure of the first substage to extend φ_e^s, $\varphi_j(x)$ does not converge. Hence, $f \notin EX(M)$. ☒

It remains an open question whether or not $Q_2EX_0[\star] - EX \neq \emptyset$.

COROLLARY 7. $EX \subset Q_1EX[\star]$.

Figure 3 summarizes all the results in this paper about query inference using a reasonable language.

$$\begin{array}{ccccc} Q_1EX_0[\star] & \subseteq & EX & = & Q_0EX[\star] \\ & & \cup\,| & & \\ Q_1EX_0[\star] & \subseteq & Q_1EX_1[\star] & & \end{array}$$

Figure 3 Summary of $[\star]$

V. Asking questions using plus and times

The ability to ask questions with plus and times is very powerful. The following theorem of Davis, Robinson, Putnam and Matijasevič [18,19,29] will be used to show that $\mathcal{R}$, the set of recursive functions, is a member of $Q_1EX[+,\times]$.

THEOREM 8. [18,29] If A is an r.e. set, then there exists a polynomial p such that

$$x \in A \Leftrightarrow \exists \vec{z}\,[p(\vec{z}, x) = 0].$$

Moreover, given a program e, the polynomial p corresponding to the domain of φ_e can be found effectively.

PROPOSITION 9. $\mathcal{R} \in Q_1EX[+, \times]$.

Proof: Using the language $[+, \times]$ questions of the type "$\exists \vec{z}\,[p(\vec{z}) = 0]$?" can be formulated for an arbitrary polynomial p. By Theorem 8, such questions can be used to solve the halting problem. Next we describe the operation of a QIM M that can infer all the recursive functions using knowledge about solutions to the halting problem. Suppose M is trying to learn a recursive function f. Initially, M conjectures program 0. Then M asks questions to find the values of $f(0)$, $f(1)$, $\cdots$ (by Proposition 4) and while doing so checks to see if $\varphi_0(x)$ is defined (requiring queries to the halting problem) and is equal to $f(x)$. If not, then program 1 is conjectured and tested similarly. This process continues, testing each program in turn. Suppose e is the least index such that $\varphi_e = f$. Then programs 0 through $e - 1$ will be rejected by M. M will then converge to program e. ☒

The machine M in the above proof uses a modification of the enumeration technique due to Gold [24]. Gold's technique only serves to infer effectively enumerable sets of recursive functions. Starting with an enumeration of programs for the functions to be inferred, Gold's technique eliminates incorrect programs one by one without having to appeal to the halting problem.

The proof of the above theorem leads to a generalization: if L is a language such that the halting set (HALT) can be solved with queries to L, then $\mathcal{R} \in QEX[L]$. Blum and Adleman [1] (see also [32]) have shown that if A is high (i.e., the halting problem relative to oracle A is Turing equivalent to the the halting problem relative to oracle HALT, in symbols

$A' \equiv_T \emptyset''$) then the problem of passively inferring a function can be solved recursively in A. Hence if L is a language such that some high set A can be solved with queries in L, then (even if A is incomplete, i.e. $A <_T \text{HALT}$) $\mathcal{R} \in QEX[L]$. We do not know of any natural languages with this property, but an artificial one can be constructed. Let A be an incomplete r.e. high set. Such A's are shown to exist in [42]. By Theorem 8 there exists a polynomial $p(z_1, \ldots, z_k, x)$ such that

$$x \in A \text{ iff } \exists z_1, \ldots, z_k [p(z_1, \ldots, z_k, x) = 0].$$

The question "$x \in A$?" is equivalent to the question "$\exists z_1, \ldots, z_k [p(z_1, \ldots, z_k, x) = 0]$?" Hence it is easy to see that $\mathcal{R} \in Q_1 EX[p]$.

Inference via the enumeration technique gives no bound on the number of mind changes required to converge to a correct program. Since $\mathcal{R} \notin EX$, by Theorem 5, $\mathcal{R} \notin Q_1 EX_0[+, \times]$. The question of whether or not $\mathcal{R} \in Q_1 EX_c[+, \times]$ is open for $c > 0$. However, if questions using an alternation of quantifiers are allowed, then all the recursive functions can be inferred by a single QIM that never changes its mind. This indicates some sort of tradeoff between mind changes and alternations of quantifiers. Other tradeoffs with the number of mind changes, in the context of anomalies and teams, have been previously noted [13,14,20,41].

THEOREM 10. $\mathcal{R} \in Q_2 EX_0[+, \times]$.

Proof: Let $\langle \cdot, \cdot \rangle$ denote the pairing function from $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$ found in [35]. The important feature of this particular pairing function is that the value of $\langle x, y \rangle$ can be expressed as a polynomial in x and y. By Theorem 8, there is an effective list of polynomials p_0, p_1, ... such that, for all i,

$$\varphi_i(x) \text{ converges to } y \Leftrightarrow \exists \vec{z} [p_i(\vec{z}, \langle x, y \rangle) = 0].$$

Let M be a QIM that asks:

$$\forall x \exists \vec{z} \, [p_0(\vec{z}, \langle x, \mathcal{F}(x) \rangle) = 0]$$

$$\forall x \exists \vec{z} \, [p_1(\vec{z}, \langle x, \mathcal{F}(x) \rangle) = 0]$$

$$\vdots$$

$$\forall x \exists \vec{z} \, [p_i(\vec{z}, \langle x, \mathcal{F}(x) \rangle) = 0]$$

until (if ever) the answer is YES. If the question that produced the YES answer involved p_i, then M outputs i and halts. Clearly, $\mathcal{R} \in Q_2EX_0[+, \times](M)$. ☒

Figure 4 summarizes all the results in this paper about query inference using the language $[+, \times]$.

$$EX = Q_0EX[+, \times] \subset Q_1EX[+, \times] = \begin{array}{c} \mathcal{R} \in Q_2EX_0[+, \times] \\ \| \\ Q_2EX[+, \times] \\ \| \\ Q_3EX[+, \times] \\ \| \\ \vdots \\ \| \end{array}$$

Figure 4 Summary of $[+, \times]$

VI. Asking questions using plus and less than

The ability to ask quantified questions using plus and less than allows the inference of sets not in EX, but not the set of all recursive functions. The language $[+, <]$ is based on Presburger arithmetic.

PROPOSITION 11. $EX \subset Q_1EX[+, <]$.

Proof: By Corollary 7 with $[\star] = [+, <]$. ☒

The set $S \in Q_1EX_1[+, <] - EX$, given by Theorem 6, is inferred with one mind change. This cannot be improved since, by Theorem 5, $Q_1EX_0[+, <] \subseteq EX$. However, if we allow an alternation of quantifiers, then the S of Theorem 6 can be inferred with no mind changes.

THEOREM 12. $Q_2EX_0[<] - EX \neq \emptyset$, hence $Q_2EX_0[+,<] - EX \neq \emptyset$

Proof: Let S be as in Theorem 6. By the proof of Theorem 6, $S \notin EX$. Hence, it suffices to show that $S \in Q_2EX_0[+,<]$. For this purpose we describe a QIM M. Suppose $f \in S$. First, M mimics the QIM of Theorem 6, and finds a value e that appears twice in the range of f. Unlike the QIM of Theorem 6, M does not output this e. Instead, M asks:

$$\exists x \forall y [y \geq x \Rightarrow \mathcal{F}(y) = e]?$$

If the answer is NO then output e. If the answer is YES then ask:

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

$$\forall x [x \geq 1 \Rightarrow \mathcal{F}(x) = e]?$$

$$\vdots$$

$$\forall x [x \geq b \Rightarrow \mathcal{F}(x) = e]?$$

until a b is found that produces a YES answer. Using the techniques of Proposition 4, discover $\{f(0), f(1), \ldots, f(b)\}$ and output a program for the function

$$g(x) = \begin{cases} f(i) & \text{if } i \leq b, \\ e & \text{otherwise.} \end{cases}$$

$\boxtimes$

Subsequent to our work, it has been shown that $EX - QEX_0[+,<] \neq \emptyset$ [23], so $EX - Q_2EX_0[+,<] \neq \emptyset$. Hence, EX and $Q_2EX_0[+,<]$ are incomparable.

THEOREM 13. $\mathcal{R} \notin Q_1EX[+,<]$.

Proof: 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 effective stages of finite extension. 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, $f^0 = \emptyset$. At every stage, a default function, h, is constructed. It will turn out that either f or one of the default functions will be a recursive function not in $Q_1EX[+,<](M)$. 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 $e = g(M(b_1 b_2 \ldots b_m))$, 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 [14]. In addition, a default function is constructed in case these efforts are not successful. The construction of the default function below is more intricate than usual.

Let $\vec{b} = \langle b_1, \ldots, b_m \rangle$. 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. Initialize the default function by setting $h = f^s$. In a similar fashion, h will also be extended without indexing. Simultaneously execute the following two substages.

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

Substage 2. Force a mind change or extend the default function. 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 3 to effectively find out if there is a finite sequence τ_i extending h and an $n \in \mathbb{N}$ such that $\langle \tau_i, n \rangle \models \psi_i$. By Lemma 2, for some $i \in \{0,1\}$, τ_i exists.

If there exists $i \in \{0,1\}$ such that $g(M(\vec{b}i)) \neq e$ 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 $\vec{b} = \vec{b}i$, $h = \tau_i \cup \{(z,0)\}$, for z the least number not in the domain of τ_i, and repeat substage 2 for these new values of h and $\vec{b}$.

End stage s.

If every stage of the construction terminates, then $f(x)$ is defined for all x and hence f is a recursive function. If substage 2 causes termination of infinitely many stages, then M outputs infinitely many changes of conjecture and hence does not converge. In this case, M cannot infer f. If M converges to e 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, h eventually becomes defined on every argument. We show that M does not infer h. Let $\vec{b}$ denote the value of $\vec{b}$ on entry into stage s and let $e = g(M(\vec{b}))$. By the failure of substage 2 to terminate stage s, when M tries to infer h, it will converge to e. By the failure of substage 1 to terminate stage s, $\varphi_e(x^s)$ is undefined. Hence, $h \notin Q_1EX[+,<](M)$. ☒

The only property of $[+,<]$ that is used in the proof is that the existential theory of $[+,<]$ is decidable. (This was used while invoking Lemma 3 during substage 2.) Hence we have the following generalization: if L is any language such that the theory of existential sentences associated with L is decidable, then $\mathcal{R} \notin Q_1EX[L]$. A natural example of such a language is $[+,D]$, where D is the three place predicate $D(x,y,z)$ which is true iff z is the greatest common divisor of x and y [28].

The notions of anomalies, behaviorally correct inference [14], and teams [41], can be naturally combined with our notion of query inference. Once these notions are understood, it is easy to define the classes $[1,n]Q_iBC^a[L]$. Using the techniques of Theorem 13 with those of the above referenced papers it is possible to show that for all $n, a \in \mathbb{N}$, $\mathcal{R} \notin [1,n]Q_1BC^a[+,<]$. Subsequent to our work, using additional techniques, it has been shown that $\mathcal{R} \notin QEX[+,<]$ [23]. In fact, these new techniques can be used to show $\mathcal{R} \notin [1,n]QBC^a[+,<]$.

Figure 5 summarizes all the results in this paper about query inference using the language $[+,<]$.

$$EX = Q_0EX[+,<] \subset Q_1EX[+,<]$$

$$\mathcal{R} \notin Q_1EX[+,<]$$

Figure 5 Summary of $[+,<]$

VII. Asking questions involving successor and less than

In this section we examine QIM's that are trying to infer functions by asking questions in the language whose only additional symbols are S (for successor) and $<$ (for less than), i.e. in the language $[S,<]$. This language is based on the decidable second order theory $S1S$ (second order theory of one successor) which contains the symbol S. The symbol $<$ can be written in terms of S using second order variables. We include $<$ explicitly and do not use second order variables. The ability to ask questions using successor and less than allows the inference of functions not in EX, but not the set of all recursive functions.

PROPOSITION 14. $EX \subset Q_1EX[S,<]$.

Proof: By Corollary 7 with $[\star] = [S,<]$. □

The above theorem indicates that allowing a quantifier gives enough power to the query language $[S,<]$ to infer all sets in EX, and then some. Next we will show that if the QIM is restricted to output a bounded number of conjectures, then not even unbounded quantifier alternations will enable the inference of all sets in EX.

For the rest of this section we only consider the inference of sets. This convention will not lead to weaker theorems (with one exception noted below) and will allow us to use the theory of ω-automata. The language is adjusted as follows. The variable $\mathcal{X}$ will be used to denote the set that some QIM is trying to infer. As a consequence, we will write "$t \in \mathcal{X}$" ("$t \notin \mathcal{X}$") instead of "$\mathcal{F}(t) = 1$" ("$\mathcal{F}(t) = 0$"). Every question asked using this language is a formula in the Second Order Theory of one successor. This theory is commonly referred to as $S1S$.

The proofs of most theorems in this section depend on the relationship between $S1S$ and ω-automata theory. This relationship is reviewed before presentation of the main result. If $V \subseteq \{0,1\}^\star$ then let V^ω be the set of all infinite sequences of elements chosen from V. If $N = \langle Q, \Sigma, \Delta, s, F\rangle$ is a nondeterministic finite automaton and $B \in \{0,1\}^\omega$, then N *accepts* B if there exists some computation path that visits members of Finfinitely often. F is called the set of *favorable* states. The set of all $B \in \{0,1\}^\omega$ accepted by N is denoted by $L(N)$. A set $A \subseteq \{0,1\}^\omega$ is *ω-regular* if there exists a nondeterministic finite automaton N such that $L(N) = A$. There are many equivalent definitions [11,16,30,36] of ω-regular languages. Büchi [11] showed that the class of ω-regular languages is closed under union, intersection, and complementation. If $A \subseteq \mathbb{N}$, then the *characteristic sequence of* A is $a_0 a_1 a_2 \ldots$ where $a_i = 0$ if $i \notin A$ and $a_i = 1$ if $i \in A$.

By representing sets of natural numbers via their characteristic sequences we can think of an ω-regular language as a class of sets. On occasion, we will unambiguously identify a set with its characteristic sequence. Suppose ψ is a formula in $S1S$ with one free set variable ($\mathcal{X}$) and no other free variables of any kind. If U is a set, $U \models \psi$ means that ψ is true when $\mathcal{X}$ is interpreted as U. Formally, the definition of $U \models \psi$ is similar to that of $f \models \psi$ in section III. Büchi [11] linked $S1S$ with ω-regular languages as follows.

THEOREM 15. [11] If ψ is a formula in $S1S$ with one free set variable and no other free variables of any kind, then the set $\{U \,|\, U \models \psi\}$ is ω-regular. Moreover, there is an effective procedure to pass from ψ to the nondeterministic finite automaton.

Suppose that ψ is a formula within the scope of Theorem 15. The set $\{U \,|\, U \models \psi\}$ is denoted by $L(\psi)$ and $N(\psi)$ is the nondeterministic finite automaton that accepts the ω-regular language $L(\psi)$. If N and N' are two nondeterministic finite automata, then $N \cap N'$ denotes the nondeterministic finite automaton accepting $L(N) \cap L(N')$. Büchi [11] also showed that one can effectively test for emptiness and infiniteness of an ω-regular language presented as a nondeterministic finite automaton.

A string $x \in \{0,1\}^\omega$ is *periodic* if there exist $u, v \in \{0,1\}^\star$ such that $x = u(v)^\omega$. A set is *periodic* if its characteristic string is periodic. We extend the definition of primitive recursiveness to infinite strings by defining a string $x \in \{0,1\}^\omega$ to be primitive recursive if its characteristic set is primitive recursive. We use the term "string" to refer to both finite and infinite strings; the meaning will be clear from context.

LEMMA 16. If V is a regular set such that $|V^\omega| \geq 2$, then

a) $V^\star$ contains two distinct strings y and z of the same length,

b) V^ω is uncountable, and

c) V^ω contains an infinite number of nonperiodic primitive recursive strings.

Proof: Let w and x be two distinct strings of V^ω. Since $w \neq x$, there exists $w', x' \in V^\star$ such that w' is a prefix of w, but not of x, and x' is a prefix of x, but not of w. Let

$$y = \overbrace{w'w' \cdots w'}^{|x'| \text{ times}}$$

$$z = \overbrace{x'x' \cdots x'}^{|w'| \text{ times}}$$

The strings y and z satisfy *a*). The set $\{y, z\}^\omega$ is an uncountable subset of V^ω, so V^ω is uncountable. The strings

$$yzyzzyzzzy \ldots$$

$$yyzyzzyzzzy \ldots$$

$$\vdots$$

are all nonperiodic primitive recursive elements of V^ω. ☒

LEMMA 17. Every ω-regular set that contains a nonperiodic string contains an infinite number of nonperiodic primitive recursive strings.

Proof: Every nonempty ω-regular set is of the form

$$\bigcup_{i=1}^{m} U_i \cdot (V_i)^\omega$$

where $m \in \mathbb{N}$ and the U_i's and V_i's are nonempty, ϵ-free regular sets [16]. Suppose w is a nonperiodic string in the ω-regular set depicted above. Assume without loss of generality that $w \in u \cdot (V_1)^\omega$ where $u \in U_1$. Choose $v \in V_1$ and let $x = u \cdot (v)^\omega$. Since w is nonperiodic and x is periodic, w and x are two distinct elements of $u \cdot (V_1)^\omega$. Hence, V_1^ω has two distinct strings. By Lemma 16, V_1^ω contains an infinite number of nonperiodic primitive recursive strings, so $u \cdot (V_1)^\omega$ does as well. $\square$

THEOREM 18. For any $a \in \mathbb{N}$, $EX - QEX_a[S, <] \neq \emptyset$.

Proof: By the enumeration technique, the set of all primitive recursive sets is in EX [24]. We show that if M is a QIM using the query language $[S, <]$ that changes its conjecture at most a times, then there exists a primitive recursive set that M does not infer. The proof is nonconstructive. We start with the set of all primitive recursive sets and remove members that M may infer, and show that a nonempty set remains. Let $\vec{b}_0 := \lambda$ (the empty string), $e_0 :=\perp$ (the null guess) and $\mathcal{A}_1 :=$ the set of all primitive recursive sets.

For $i := 1$ to $a + 1$ do

$A_i :=$ some primitive recursive set in $\mathcal{A}_i$ that is not periodic. (Later we will show that $\mathcal{A}_i$ always contains such a set.)

$\vec{b}_i :=$ the least element of $\{0,1\}^\star$ (if it exists) such that $\vec{b}_i$ extends $\vec{b}_{i-1}$, $M(\vec{b}_i) \neq M(\vec{b}_{i-1})$, and $\vec{b}_i$ supplying correct information about A_i in reply to M's queries.If there is no such vector, then $\vec{b}_i = \vec{b}_{i-1}$. (This step is nonconstructive.)

$\Psi_i :=$ the conjunction of all the formulas that the run of $M(\vec{b}_i)$ discovers A_i satisfies. Formally, Ψ_i is the conjunction of the formulas ψ'_1, ψ'_2, ..., ψ'_s, where s is the length of $\vec{b}_i$ and ψ'_1, ψ'_2, ..., ψ'_s are defined as follows:

For $j := 1$ to s, if $g(M(b_1 \ldots b_{j-1})) = \perp$

then $\psi_j := q(M(b_1 \ldots b_{j-1}))$

$\psi'_j := \psi_j$ if $b_j = 1$, $\neg\psi_j$ if $b_j = 0$;

else $\psi'_j :=$ TRUE.

$N_i :=$ $N(\Psi_i)$ Note that N_i is the ω-automaton that accepts the intersection of the ω-regular languages associated with the formulas $\psi'_1, \psi'_2, \ldots, \psi'_s$. Since N_i accepts the set A_i, and A_i is nonperiodic, by Lemma 17, N_i accepts an infinite number of nonperiodic primitive recursive sets.

$e_i :=$ $g(M(\vec{b}_i))$.

$\mathcal{A}_{i+1}:=$ $(\mathcal{A}_i \cap L(N_i)) - X$ where $X = \emptyset$ if φ_{e_i} is not total or not $\{0,1\}$ valued, and X is the singleton set containing the infinite string that represents the set associated with function φ_{e_i} otherwise. (Later we show that $\mathcal{A}_{i+1}$ contains an infinite number of nonperiodic primitive recursive sets.)

It is easy to see that for all i, $L(N_i) \subseteq L(N_{i-1})$, and $L(N_i)$ accepts an infinite number of primitive recursive strings. An induction argument shows that each of the $\mathcal{A}_{i+1}$'s contain all but a finite number of the nonperiodic primitive recursive strings in N_i. Hence, each of the $\mathcal{A}_i$'s contain an infinite number of nonperiodic primitive recursive strings.

The class $\mathcal{A}_i$ is defined such that the answers given to M do not distinguish any set in $\mathcal{A}_i$ from any other set in $\mathcal{A}_i$. Furthermore, M's current conjecture is not a set in $\mathcal{A}_i$. Hence, the set A_i satisfies all the answers given to M thus far and is not the set currently conjectured by M. If $\vec{b}_i = \vec{b}_{i-1}$ then M has stopped changing its conjecture and A_i is the desired set that M cannot infer. 𝕏

The set of primitive recursive sets is contained in a strict subset of EX called PEX, where all conjectures must be programs computing total functions [7]. Hence, the above proof actually shows that, for any $a \in \mathbb{N}$, $PEX - QEX_a[S,<] \neq \emptyset$. By Theorem 6 $Q_1EX_1[S,<] - EX \neq \emptyset$. Combining this result with Theorem 18 we obtain the following.

THEOREM 19. For all $a > 0$, $QEX_a[S,<]$ and EX are incomparable.

If we are restricted to inferring sets, then this is not true for the $a = 0$ case. The analogous question for functions remains open.

THEOREM 20. When restricted to sets, $QEX_0[S,<] \subset EX$.

Proof: Let M be a QIM using the query language $[S,<]$. We define an IIM M' such that $QEX_0[S,<](M) \subset EX(M')$. M' simulates M's behavior on some set A assumed to be in $QEX_0[S,<](M)$. The simulation is carried out in effective stages. At stage $s > 0$, a sequence $\vec{b} \in \{0,1\}^s$ is sought such that $\vec{b}$ "approximates" the correct sequence of answers to the first s queries made by M while inferring A. Recall that in the proof of Theorem 5 we also had to look for an appropriate $\vec{b}$.

Stage s: First we determine, for each sequence $\vec{b} = b_1 \ldots b_s \in \{0,1\}^s$, what the implications of using $\vec{b}$ as input to M are. Define $\psi'_1, \ldots, \psi'_s$ as follows:

For $i := 1$ to s, if $g(M(b_1 \ldots b_{i-1})) = \perp$

then $\psi_i := q(M(b_1 \ldots b_{i-1}))$

$\psi'_i := \psi_i$ if $b_i = 1$, $\neg\psi_i$ if $b_i = 0$;

else $\psi'_i :=$ TRUE.

Let $N_{\vec{b}}$ be a nondeterministic finite automaton (interpreted as an ω-automaton) that recognizes all strings in $\{0,1\}^\omega$ that satisfy $\psi'_1 \wedge \cdots \wedge \psi'_s$. M' accepts data until it has as input σ, the length s initial segment of A. Run $N_{\vec{b}}$ on σ, following all possible nondeterministic paths. Note which path maximizes the sum total of all visits to all favorable states. Associate this maximum number with $\vec{b}$ and call it $num_{\vec{b}}$. Choose $\vec{b}$ with $num_{\vec{b}}$ largest. In case of a tie, choose the lexicographically least qualifying $\vec{b}$. Output $g(M(\vec{b}))$.

End stage s.

The proof is completed by showing that the set A is inferred in the limit by M'. Since M does not change its mind, there exists a finite sequence $\vec{b}$ such that (a) $g(M(\vec{b})) = e$, a program for A, (b) $g(M(\vec{b}')) = e$ for all $\vec{b}'$ extending $\vec{b}$, and (c) $\vec{b}$ supplies the correct

answers to questions posed by M about A. There is a constant c such that for all $\vec{b}'$ different from $\vec{b}$, but with the same length, $N_{\vec{b}'}$ on input A never visits a favorable state more than c times. Let s be such that $N_{\vec{b}}$ on the initial segment of A of length s visits a favorable state more than c times. Such an s exists because num is increasing with s. For all $t > s$, M' at stage t will output e. Hence, $A \in EX(M')$. ☒

Theorem 5 and Theorem 20 are similar in that both involve approximating the correct answers. A general theorem could be stated that has both theorems as corollaries, but we do not pursue this in the present paper.

LEMMA 21. The problem of testing if an arbitrary ω-regular language is uncountable is decidable.

Proof: By [16], given any of the standard descriptions of an ω-regular language L, one can effectively find nonempty, ϵ-free regular sets such that

$$L = \bigcup_{i=1}^{m} U_i \cdot (V_i)^{\omega}.$$

For every i, determine if $V_i^{\star}$ contains two distinct strings, y and z, of the same length. If for some i, such a y and z exist, then $U_i \cdot \{y, z\}^{\omega} \subseteq L$, so L is uncountable. If not, then by part a) of Lemma 16, for all i $(1 \leq i \leq m)$, $|V_i^{\omega}| \leq 1$, so L is countable. ☒

In the following lemma, we consider the behavior of NDFA's on finite prefixes of infinite strings. A NDFA *accepts* a finite string σ, if when started in the initial state and receiving σ as input, the NDFA ends up in a favorable state.

LEMMA 22. Suppose N is an NDFA such that the ω-regular language $L(N)$ is uncountable. Then there exists a finite string σ such that the subsets of strings from $L(N)$ with prefix $\sigma 0$ and $\sigma 1$ are both uncountable, and there exists τ, a prefix of σ, such that N accepts τ as a finite string.

Proof: Suppose the hypothesis. Let D be all the finite strings that N accepts (as an NDFA) without repeating a state on the accepting path. For some $\tau \in D$, $\tau \cdot \{0,1\}^\omega \cap L(N)$ is uncountable. Suppose by way of contradiction that for every finite string σ that extends τ one of $\sigma 0\{0,1\}^\omega \cap L(N)$ or $\sigma 1\{0,1\}^\omega \cap L(N)$ is countable. For $b \in \{0,1\}$, $\bar{b}$ denotes $1-b$. Choose b_1 such that $U_1 = \tau b_1\{0,1\}^\omega \cap L(N)$ is countable. Inductively define $b_m \in \{0,1\}$ such that $U_m = \tau \bar{b}_1 \bar{b}_2 \cdots \bar{b}_{m-1} b_m \{0,1\}^\omega \cap L(N)$ is countable. Let w denote the infinite string $\tau \bar{b}_1 \bar{b}_2 \cdots$. If $w \in \tau \cdot \{0,1\}^\omega \cap L(N)$ then

$$\tau \cdot \{0,1\}^\omega \cap L(N) = \bigcup_{m=1}^{\infty} U_m \cup \{w\}.$$

Otherwise,

$$\tau \cdot \{0,1\}^\omega \cap L(N) = \bigcup_{m=1}^{\infty} U_m.$$

In either case, $\tau \cdot \{0,1\}^\omega \cap L(N)$ is countable, a contradiction. ☒

THEOREM 23. $QEX[S,<]$ does not contain the set of all recursive languages.

Proof: Let M be a QIM that asks questions using the language $[S,<]$. We construct, in effective stages, a recursive set $A \notin QEX[S,<](M)$. At the beginning of every stage s we will have:

1) $A_s \in \{0,1\}^\star$, an initial segment of A.
2) Formulae ψ_1, ψ_2, ..., ψ_m of $S1S$ with one free set variable and no free number variables and bits b_1, b_2, ..., $b_m \in \{0,1\}$ such that for all i, $1 \leq i \leq m$, ψ_i is the i^{th} question asked by M, and b_i is its answer. (Our intention is that for $1 \leq i \leq m$, $b_i = 1 \Leftrightarrow A \models \psi_i$, and $b_i = 0 \Leftrightarrow A \models \neg\psi_i$.)
3) A nondeterministic finite automaton N_s which (i) accepts an uncountable subset of $A_s \cdot \{0,1\}^\omega$ and $(ii)$for every $B \in \text{L}(N_s)$,
   a) $b_i = 1 \Rightarrow B \models \psi_i$, and
   b) $b_i = 0 \Rightarrow B \models \neg\psi_i$.

The strategy of the construction is to extend A_s in such a way that N_s will accept A. This will mean that when A is constructed all the answers M received in response to its queries are consistent with A. The crucial aspect of the use of ω-automata theory is to restrict the as yet unspecified part of A *without* determining it precisely. We make implicit use of Lemma 21 and Lemma 22.

Begin construction of A. Initialize by setting $A_0 = \lambda$, the empty string. Let N_0 be a nondeterministic finite automaton accepting $\{0,1\}^\omega$. Execute the following effective stages in their natural order.

Stage s. Let x be the least number whose membership in A is not yet specified. Suppose that so far in the construction M has asked m questions and received answers b_1, b_2, ..., b_m. Let e be the most recent conjecture by M as to a program to decide membership in A.

Remark: Most diagonalizations in inductive inference attempt to extend the construction by making the current guess wrong or by forcing a mind change. If neither of these can be accomplished, some default extension suffices. These components are also present here. $\varphi_e(x)$ must be defined in order to decide whether to place x in A or to keep x out of A. Finding an A_{s+1} that forces M to change its mind entails answering another question that may influence infinitely many values potentially in A. Hence, another automaton, N_{s+1} will be determined. This device is used to guarantee that the set constructed is consistent with the answers given to M about the nature of A. The construction of the default set proceeds simultaneously with the simulation of $\varphi_e(x)$ and the search for a mind change.

Suppose ρ is a finite string of 0's and 1's (representing a finite set) and N is a nondeterministic finite automaton. We define $EXT(\rho, N)$ to be σ, the lexicographically least string of 0's and 1's (if it exists) such that:

1) σ is a proper extension of ρ, and

2) $L(N) \cap (\sigma \cdot 0 \cdot \{0,1\}^\omega)$ is uncountable, and

3) $L(N) \cap (\sigma \cdot 1 \cdot \{0,1\}^\omega)$ is uncountable, and

4) There exists τ such that ρ is a prefix of τ, τ is a prefix of σ and N accepts τ as a finite string.

By Lemma 22, if $L(N) \cap \rho \cdot \{0,1\}^\omega$ is uncountable, then $EXT(\rho, N)$ is guaranteed to exist. Whenever we use $EXT(\rho, N)$ it will be the case that $L(N)$ is uncountable and $L(N) \subseteq \rho \cdot \{0,1\}^\omega$, so $EXT(\rho, N)$ will exist.

Let $\vec{b} = \langle b_1, b_2, \ldots, b_m \rangle$. This vector of responses will be lengthened during this stage. To reduce notation, the various, larger and larger, vectors will not be indexed. Therefore, $\vec{b}$ always denotes the *current* vector. Initialize the default set B by setting $B_m = A_s$. Let $M(\vec{b}) = \langle e, \psi \rangle$.

The following three substages are iterated for $t = m+1, m+2, \ldots$ until (if ever) conditions for commencing the next stage are satisfied.

Substage 1. Diagonalize on x. If $\varphi_e(x)$ is not convergent in $\leq t$ computation steps then go to Substage 2. Otherwise, let $\psi = q(M(\vec{b}))$ and let $diag$ denote the $S1S$ formula "$X(x) = 1 \dot{-} \varphi_e(x)$."

If $L(N_s \cap N(diag) \cap N(\psi))$ is uncountable then

$N_{s+1} := N_s \cap N(diag) \cap N(\psi)$,

$A_{s+1} := EXT(A_s, N_{s+1})$,

$b_{m+1} := 1$,

$\vec{b} := b_0 b_1 \ldots b_{m+1}$, and

Go to stage $s+1$.

Else

$N_{s+1} := N_s \cap N(diag) \cap N(\neg\psi)$,

$A_{s+1} := EXT(A_s, N_{s+1})$,

$b_{m+1} := 0$,

$\vec{b} := b_0 b_1 \ldots b_{m+1}$, and

Go to stage $s+1$.

Substage 2. Find a mind change. If we find a mind change (details below) then go to stage $s+1$ as indicated. If not, go to substage 3. Let $\psi_t = q(M(\vec{b}))$. The formulae ψ'_i, $m+1 \leq i \leq t-1$, are defined in the next substage before their use below. Define

$$\hat{N} = N(\psi'_{m+1} \wedge \cdots \wedge \psi'_{t-1} \wedge \psi_t)$$

$$\overline{N} = N(\psi'_{m+1} \wedge \cdots \wedge \psi'_{t-1} \wedge \neg\psi_t).$$

(If $t = m+1$ then $\hat{N} = N(\psi_t)$ and $\overline{N} = N(\neg\psi_t)$.)

If $g(M(\vec{b}1)) \neq e$ and $L(N_s \cap \hat{N})$ is uncountable then

$N_{s+1} := N_s \cap \hat{N}$,

$A_{s+1} := EXT(A_s, N_{s+1})$,

$\vec{b} := \vec{b}1$, and

Go to stage $s+1$.

If $g(M(\vec{b}0)) \neq e$ and $L(N_s \cap \bar{N})$ is uncountable then

$N_{s+1} := N_s \cap \bar{N}$,

$A_{s+1} := EXT(A_s, N_{s+1})$,

$\vec{b} := \vec{b}0$, and

Go to stage $s+1$.

Substage 3. Extend the default set B. Let $\hat{N}$ and $\bar{N}$ be as in Substage 2.

If $L(N_s \cap \hat{N})$ is uncountable then

$B_t := EXT(B_{t-1}, N_s \cap \hat{N})$,

$\psi'_t := \psi_t \wedge \bigwedge_{i=0}^{|B_t|} (X(i) = B(i))$, and

$\vec{b} := \vec{b}1$

Else

$B_t := EXT(B_{t-1}, N_s \cap \bar{N})$,

$\psi'_t := \neg\psi_t \wedge \bigwedge_{i=0}^{|B_t|} (X(i) = B(i))$, and

$\vec{b} := \vec{b}0$.

End Stage s.

End Construction.

If every stage s of the construction terminates, then every x is placed either in or out of A. Hence A is recursive. Moreover, if M's sequence of output programs converges then by substage 1, M's final conjecture is incorrect infinitely often. Substage 1 could force M to never converge. In either case, M does not infer A.

If some stage s never terminates, then substage 3 of stage s creates a recursive set B. Let $\vec{b}'$ denote the value of $\vec{b}$ on entry into stage s. By the failure of substage 2 to terminate stage s, M when attempting to infer B will converge to $e = g(\vec{b}')$. By the failure of substage 1 to terminate stage s, $\varphi_e(x)$ is undefined for some x, hence M cannot infer B. ☒

As noted in the discussion of results concerning the query language $[+, <]$, the notions of anomalies, behaviorally correct inference 14, and teams 41, can be naturally combined with our notion of query inference. Once these notions are understood, it is easy to define the class $[1, n]Q_iBC^a[L]$. Using the techniques of Theorem 23 with those of the above referenced papers it is possible to show that for all $n, a \in \mathbb{N}$, $\mathcal{R} \notin [1, n]QBC^a[S, <]$.

The language that allows second order quantifiers and $[S, <]$ is so powerful that two quantifiers suffice to ask any question in any "ω-automata" type language.

THEOREM 24. $QEX[2^{\text{nd}}, S, <] = Q_2EX[2^{\text{nd}}, S, <]$.

Proof: By [12] all ω-regular languages can be written as the set of solutions to a two quantifier formula in the language $[2^{\text{nd}}, S, <]$. ☒

Figure 6 summarizes all the results in this paper about query inference using the language $[S, <]$.

$$QEX_0[S, <] \subset EX = Q_0EX[S, <] \subset Q_1EX[S, <] \subseteq QEX[S, <] \subset \mathcal{R}$$

$$\cap \cup$$

$$QEX_a[S, <], a > 0$$

Figure 6 Summary of $[S, <]$

VIII. Conclusions

We have examined machines that learn by asking questions and have compared this notion to that of machines that learn by passively examining data. The intuition that asking questions aids the learning process was formally verified in a recursion theoretic model. Our results corroborate Angluin's [5] in that we found that the language used to formulate queries has great impact on what can be learned. The undecidable language $[+, \times]$ was found to be too powerful for use as a query language. The language $[S, <]$ is somewhat limited in what it can express as evidenced by Theorem 20. The most interesting query language for future investigations is $[+, <]$.

Several open problems remain. For the languages discussed in this paper is there an infinite hierarchy of learnability based on alternation of quantifiers? For $[+, <]$, we believe so. In light of Theorem 24 we believe not for $[S, <]$. For the base language, if recursive sets are being inferred, $QEX[\,] = Q_1EX[\,] \neq EX$ [23]. What about other languages? It would also be of interest to look at anomalies, mind changes [14], teams [41], and behavioral correctness [14] in this context. Some preliminary results have been obtained [22].

Recall that $Q_1EX_0[\star] \subseteq EX$. In this paper we proved that $Q_1EX_0[S, <] \subset EX$ by showing that the primitive recursive sets are not in $Q_1EX_0[S, <]$. Furthermore, as shown in [23], the primitive recursive sets are not in $Q_1EX_0[+, <]$, hence $Q_1EX_0[+, <] \subset EX$. The question "Is $Q_1EX_0[+, \times]$ *properly* contained in EX?" is particularly interesting since the primitive recursive sets are in $Q_1EX_0[+, \times]$. Determining the exact relationship between EX and $Q_1EX_0[+, \times]$ may require fundamental insights into the nature of EX along the lines of [21].

As mentioned in the technical summary, the teacher answering the questions of a QIM may require an oracle for an undecidable set. This undecidable set may depend on the function being inferred and the query language. For example, questions in $[+, \times]$ using i alternations of quantifiers will require at least a Σ_i-complete oracle, perhaps stronger, depending on the function to be inferred. Even with the base language there are recursive

functions f such that answering existential questions about f in this language requires the halting set. For example, if the range of f is the halting set, then queries of the form "$\exists x[\mathcal{F}(x) = a]$" require knowing whether or not a is in the halting set. Characterizing the needed oracle strength, in terms of the query language and function to be inferred, remains an open problem.

A possible interpretation of our results is that a more powerful query language allows more complicated questions and enhances learning ability qualitatively. If the human brain is a computer, albeit biological, then our results apply to human learning as well. Of course, at least in some cases, for humans to take advantage of learning enhancement by queries, they would need access to question answering oracles surpassing the computable. Nonetheless, one interpretation of our results is that increased ability to articulate questions enhances learning potential. The results concerning questions with quantifiers compared with quantifierless questions points out the utility of being able to ask questions about a phenomenon's global behavior rather than its local behavior.

IX. Acknowledgements

Conversations with Manuel Blum and John Case enhanced this document. Mark Pleszkoch suggested several improvements in the exposition, and helped with the proofs of Lemma 16 and Lemma 17. Mark Berman, John Guthrie, Valentina Harizanov, Kathleen Romanik and V. S. Subrahmanian made insightful useful comments on an earlier draft. Larry Herman assisted us in the investigation of ω-automata theory. The referees made several valuable comments that resulted in improvements to this paper.

References

1. Adleman, L. and Blum, M. *Inductive inference and unsolvability.* Department of Electrical Engineering and Computer Science and the Electronics Research Laboratory, University of California at Berkeley, 1975.

2. Angluin, D. Learning k bounded context-free grammars. Department of Computer Science TR-557, Yale University, New Haven, CT, 1987.
3. Angluin, D. Learning regular sets from queries and counterexamples. *Information and Computation 75,* 2 (1987), 87–106.
4. Angluin, D. Learning k-term DNF formulas using queries and counter-examples. Department of Computer Science TR-559, Yale University, New Haven, CT, 1987.
5. Angluin, D. Queries and concept learning. *Machine Learning 2* (1988), 319–342.
6. 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.
7. Angluin, D. and Smith, C. H. Inductive inference: theory and methods. *Computing Surveys 15* (1983), 237–269.
8. Angluin, D. and Smith, C. H. Inductive inference. In *Encyclopedia of Artificial Intelligence*, S. Shapiro, Ed., John Wiley and Sons Inc., 1987.
9. Berman, P. and Roos, R. Learning one–counter languages in polynomial time. 28^{th} *Annual FOCS conference* (1987), 61–67.
10. Blum, L. and Blum, M. Toward a mathematical theory of inductive inference. *Information and Control 28* (1975), 125–155.
11. Buchi, J. R. On a decision method in restricted second order arithmetic. In *Logic Methodology and Philosophy of Science*, E. Nagel et al., Ed., Stanford University Press, Stanford, 1962.
12. Buchi, J. R. and Siefkes, D. The monadic second order theory of all countable ordinals. In *Decidable Theories II, Lecture Notes in Mathematics 328*, G. Muller and D. Siefkes, Ed., Springer-Verlag, New York, 1973.
13. Case, J. and Ngomanguelle, S. Refinements of inductive inference by popperian machines. *Kybernetika* (19??). to appear.
14. Case, J. and Smith, C. Comparison of identification criteria for machine inductive inference. *Theoretical Computer Science 25,* 2 (1983), 193–220.
15. Chang, C. C. and Keisler, H. J. *Model Theory.* North-Holland, New York, New York, 1977.
16. Choueka, Y. Theories of automata on ω-tapes: a simplified approach. *Journal of Computer System Sciences 8* (1974), 117–141.
17. Daley, R. P. and Smith, C. H. On the complexity of inductive inference. *Information and Control 69* (1986), 12–40.
18. Davis, M., , H. Putnam, and Robinson, J. The decision problem for exponential diophantine equations. *Annals of Mathematics 74* (1961), 425–436.
19. Davis, M. Hilbert's 10^{th} problem is unsolvable. *American Mathematical Monthly 80* (1973), 233–269.

20. 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.

21. Fulk, M. *Robustly outdoing identification by enumeration.* Manuscript.

22. Gasarch, W., Kinber, E., Pleszkoch, M., Smith, C., and Zeugmann, T. *Learning via queries with teams and anomalies.* Manuscript.

23. Gasarch, W., Pleszkoch, M., and Solovay, R. *Learning via queries with plus and times.* manuscript.

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

25. Kearns, M., Li, M., Pitt, L., and Valiant, L. On the learnability of boolean formulae. 19th *Annual STOC conference* (1987), 285–296.

26. Littlestone, N. Learning quickly when irrelevant attributes abound. In *Proceedings of the 28th Symposium on the Foundations of Computing*, Los Angeles, CA, 1987.

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

28. Martjanov, V. I. Universal extended theories of integers. *Algebra and Logic 16* (1977), 395–404.

29. Matijasevic, Y. Enumerable sets are diophantine. *Doklady Academy Nauk. SSSR 191* (1970), 279–282. Translation in Sov. Math. Dokl. 11, 1970, pp. 354–357.

30. Mcnaughton, R. Testing and generating infinite sequences by a finite automaton. *Information and Control 9* (1966), 521–530.

31. Natarajan, B. K. On learning boolean functions. 19th *Annual STOC conference* (1987), 296–304.

32. Pleszkoch, M., Gasarch, W., Jain, S., and Solovay, R. *Learning via queries to an oracle.* Manuscript.

33. Rabin, M. O. *Automata on infinite objects and Church's problem.* American Mathematics Society, Providence, RI, 1972.

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

35. Rogers, H. Jr. *Theory of Recursive Functions and Effective Computability.* McGraw Hill, New York, 1967.

36. Safra, S. On the complexity of ω-automata. In *Proceedings of the* 29th *Symposium on the Foundations of Computer Science*, White Plains, NY, 1988.

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

38. 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.

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

40. Shepherdson, J. Negation in logic programming. In *Foundations of Deductive Databases and Logic Programming*, J. Minker, Ed., Morgan Kaufmann Piblishers, Los Altos, CA, 1988.

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

42. Soare, R. I. *Recursively enumerable sets and degrees.* Springer Verlag, New York, 1987.

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