Inferring Answers to Queries [⋆]

William I. Gasarch [a,*,1] Andrew C.Y. Lee [b]

[a] *Institute of Advanced Computer Studies and Department of Computer Science, University of Maryland, College Park, MD 20742*

[b] *Computer Science Department, University of Louisiana at Lafayette, LA 70504-1771*

Abstract

One focus of inductive inference is to infer a *program* for a function f from observations or queries about f. We propose a new line of research which examines the question of inferring the *answers to queries*. For a given class of computable functions, we consider the learning (in the limit) of *properties* of these functions that can be captured by queries formulated in a logical language L. We study the inference types that arise in this context. Of particular interest is a comparison between the learning of properties and the learning of programs. Our results suggest that these two types of learning are incomparable. In addition, our techniques can be used to prove a general lemma about query inference [19]. We show that $\mathcal{I} \subset \mathcal{J} \Rightarrow \mathrm{Q}\mathcal{I}(L) \subset \mathrm{Q}\mathcal{J}(L)$ for many standard inference types $\mathcal{I}$, $\mathcal{J}$ and many query languages L. Hence any separation that holds between these inference types also holds between the corresponding query inference types. One interesting consequence is that

$$[24,49]\mathrm{QEX}_0\,([\mathrm{Succ},<]^2) - [2,4]\mathrm{QEX}_0\,([\mathrm{Succ},<]^2) \neq \emptyset.$$

Key words: Inductive Inference, Queries, Learning, Omega Automata

[⋆] An earlier version appeared in the proceedings of COLT97

[*] Correspondence author

Email addresses: gasarch@cs.umd.edu (William I. Gasarch), cxl9999@louisiana.edu (Andrew C.Y. Lee).

URLs: http://www.cs.umd.edu/~gasarch (William I. Gasarch), http://www.louisiana.edu/~cxl9999 (Andrew C.Y. Lee).

[1] Supported in part by NSF grant CCR-9732692 and CCR-01-05413

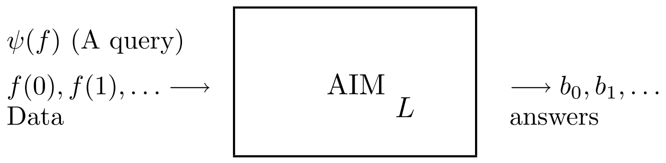
$\psi(f)$ (A query)
$f(0), f(1), \ldots \longrightarrow$
Data
AIM
L
$\longrightarrow b_0, b_1, \ldots$
answers

Fig. 1. Answer Inference Machine

1 Introduction

When scientists examine data, they may be trying to answer a question about the data (e.g. "Is the shape of the earth's orbit around the sun a circle?" [2]) or, they may be trying to find a function that fits the data (e.g. "We seek an equation that describes the earth's orbit."). In the former case we will restrict to boolean questions and hence we can think of the scientist as wondering whether a certain property holds. In the latter case we think of the scientist as trying to learn a program for the function. Note that even if the scientist has a program he might not know much about particular questions. For example, when one's looking at a program it is not decidable if it is the program for a circle.

Traditionally, an *inductive inference machine* [22,6] learns programs for computable functions. In this work we introduce a new machine model that answers logical queries about computable functions. Informally, we consider a total algorithmic device such that, when given as input a query ϕ about a function f, and an initial segment

$$\{(0, f(0)), \ldots, (n, f(n))\}$$

of f, tries to answer ϕ. We say that the machine can infer the *answer* to ϕ if it converges to the correct answer as n approaches infinity (See Figure 1).

In this model, the queries are boolean queries formulated using a logical language L. Given a language L and a concept class $\mathcal{C}$ (i.e. a collection of computable functions in our case), the collection of queries about the members of $\mathcal{C}$ that can be inferred by a machine M represents a class of *properties* of $\mathcal{C}$ that can be *learned* by M. The answer inference type, QAN (L), contains all concept classes such that boolean queries formulated in L can be inferred by such machines.

What is the difference between the learning of answers and the learning of

[2] In physics we see that Kepler's law is easier to derive than the equations of motions of the planets. Prior work motivated by these observations can be found in [7,4].

programs? In this work we show that they are *strongly* incomparable. We introduce variants of answer inference types from the machine model. They vary in their learning power. We then demonstrate the incomparability by considering the extreme cases (by comparing answer inference types with standard inference types such as PEX_0 and $[1,n]\mathrm{QBC}^a(L)$). Another theme of this paper is to demonstrate how answer inference types can be used as a technical tool in the study of query inference [19]. By examining the complexity of queries made by query inference machines, we show that separation results for many standard inference types remains true for their query counterparts. This settles a number of conjectures in query inference [19,17].

There has been some related work on this problem. Smith and Wiehagen [33] introduced a model of classification, called the *classification* machine. Given n collections of computable functions $S_1, S_2, \ldots, S_n$, a classification machine M tries to *classify* the function f in the limit by converging to some i such that $f \in S_i$. Note that a classification machine may be able to answer a *single* question in the limit but not all the questions in a given language. In their model, the S_i's are not required to be pairwise disjoint. Ben-David [5], Gasarch, Pleszkoch, Stephan and Velauthapillai [21], and Kelly [24] had studied the model where there is no limits on computational power. Kelly's work [24] includes the case when there are limits on computational power. In [21], the models consider all inputs that are elements of N^ω, including those that are *not* computable. Other works in computable classification includes [31,8]. We refer interested readers to these papers and the references cited therein on computable classification and other related topics.

The arrangement of this paper is as follows. In Section 2 we define our notations and the basic terms used in the study of answer inference types. We also state relevant definitions from Büchi automata, query languages and related concepts there. In Section 3 we summarize our technical results. Structural properties of answer inference types are discussed from Section 4 to Section 6. In Section 7 we demonstrate that the learning of programs and the learning of properties are incomparable by presenting two extreme cases. In Section 8 we further apply our techniques to prove a *lifting* lemma. It sharpens the comparison result in the previous section. It also enables us to obtain results about query inference [19] immediately from their passive versions. Concluding remarks and open problems will be stated in the last section.

2 Notations and Definitions

2.1 Standard Notations

We assume familiarity with standard definitions and notations from logic [16], computability theory [34,32] and inductive inference [3,9,23]. Throughout this work $\mathsf{N} = \{0, 1, \ldots\}$ denotes the set of all natural numbers. Given a finite set A, $|A|$ denotes the cardinality of A and A^ω denotes the set of infinite strings of length ω formed by the elements in A. $f[n]$ denotes the initial segment $\{(0, f(0)), \ldots, (n, f(n))\}$ of the graph of f. We use the symbol DEC to denote the collection of all computable functions from N to N. $\mathrm{DEC}_{0,1}$ denotes the set of all decidable subsets of N. We often think of subsets of N as $\{0, 1\}$-valued functions and interpret $\mathrm{DEC}_{0,1}$ as a subset of DEC. Subsets of DEC and $\mathrm{DEC}_{0,1}$ are referred as *concept classes* and EX, $[1, n]BC$ and etc. (resp. $\mathrm{QEX}\,(L)$, $[1, n]\mathrm{QBC}\,(L)$ ...) are referred as passive (resp. query) *inference types*.

2.2 Query Languages

A *First Order Query Language* ([19,17]) L consists of the usual logical symbols (including *equality*), symbols for number variables, symbols for every element of N, a special function symbol $\mathcal{F}$ denoting a function we wish to learn, and symbols for additional functions and relations on N. We will assume that these additional functions and relations are computable. L may be denoted by the symbols for the additional functions and relations (e.g. $[+, <]$ denotes the language with additional symbols for $+$ and $<$; $[\emptyset]$ denotes the language with no additional symbols). A well-formed formula over L is defined in the usual manner. Throughout this work, L denotes a *reasonable* query language. That is, all the symbols in L represent computable operations. The language that omits the symbol $\mathcal{F}$ from L is called the *base language* of L. We say that the base language is *decidable* if the truth of any sentences in the base language is decidable. The truth value of any sentence is either 1 (TRUE) or 0 (FALSE). We define query about functions. They are *boolean* questions that can be formulated in L. The analogous notions for sets can be defined similarly.

Definition 2.1 A *query ϕ about f* is a closed well-formed formula in L with the function symbol $\mathcal{F}$. $\phi(f)$ denotes both the question (when $\mathcal{F}$ is interpreted as f) and the correct answer ($\in \{0, 1\}$) of the question.

Note that '$\forall \phi \in L$' denotes the phrase "for every query formulated in the language L". Some examples of queries in various languages are listed in Figure 2.

L	*Example*
$[\emptyset]$	*query*: $(\forall y)(\exists x)[\mathcal{F}(x) = y]$ *interpretation*: 'Is the function surjective?'
$[<]$	*query* :$(\exists x)(\forall y)[(x < y) \to \mathcal{F}(x) = 0]$ *interpretation*: 'Does the function has finite support?'
$[+, <]$	*query*: $(\exists x)(\exists p)(\forall y)[(x < y) \wedge (p > 0) \to \mathcal{F}(y) = \mathcal{F}(y + p)]$ *interpretation*: 'Is the function eventually periodic?'

Fig. 2. Examples of queries in different languages

Consider the query language $[+, \times]$. It is known that questions to the HALTING PROBLEM can be asked in this language [14,25,13,26]. Hence the set of all computable functions can be inferred with $[+, \times]$. These questions are not about the function $\mathcal{F}$ and not really in the spirit of our inquiry. Unless otherwise stated, our primary concern will be query languages with a decidable base language.

We also use the following predicate and function symbols in some query languages.

Notation 2.2 Let $b \geq 2$. The following symbols will be used in some of our query languages:

a) Succ denotes the *successor* function Succ(x)=$x + 1$.
b) POW$_b$ is the unary predicate that determines if a number is in the set $\{b^n : n \in \mathsf{N}\}$.
c) POLY$_b$ is the unary predicate that determines if a number is in the set $\{n^b : n \in \mathsf{N}\}$.
d) FAC is the unary predicate that determines if a number is in the set $\{n! : n \in \mathsf{N}\}$.

The query language $[\text{Succ}, <]^2$ will be used frequently in this paper. Apart from the extra symbols Succ and $<$, we allow the use of set variables and their quantifications (the superscript ^{2} is used to denote this fact). We adopt the following conventions. Small (resp. Large) letters are used for number (resp. set) variables, which range over N (resp. subsets of N). Specific features of this query language will be addressed in later sections.

Definition 2.3 The *Second Order Query language* $[Succ, <]^2$ is defined as follows.

a) *Terms*: a term is either a numeric variable, a numerical constant k (interpreted as the natural number k), or is of the form $g(t)$, where t is a term and g is either the symbol Succ or $\mathcal{F}$. Note that a term does not use set variables.

b) *Atomic formulas*: any atomic formula is of the form $(t \in X)$, $(s = t)$, $(s < t)$ where s and t are terms and X is a set variable.

c) *Well-formed formulas*: well-formed formulas in the query language $[\text{Succ}, <]^2$ are defined inductively as follows:

(1) Any atomic formula is a well-formed formula.

(2) If ψ and θ are well-formed formulas, then $(\psi) \vee (\theta)$, $(\psi) \wedge (\theta)$, $(\psi) \rightarrow (\theta)$, $(\psi) \leftrightarrow (\theta)$ and $\neg(\theta)$ are well-formed formulas.

(3) If x is a numeric variable, X is a set variable and θ is a well-formed formula, then $(\exists x)[\theta(x)]$, $(\forall x)[\theta(x)]$, $(\exists X)[\theta(X)]$, and $(\forall X)[\theta(X)]$ are well-formed formulas.

(4) Nothing else is a well-formed formula.

d) *Queries*: queries are the closed well-formed formulas with the function symbol $\mathcal{F}$.

2.3 Answer Inference Types

Definition 2.4

a) An *answer inference machine* M (abbrev. AIM) is a total Turing machine such that on each input $\phi \in L$ and each initial segment $f[n]$ $(n \geq 0)$ of f, output a guess of the truth value of $\phi(f)$, which is denoted by $M(\phi, f[n]) \in \{0, 1\}$. The limit $\lim_{n\to\infty} M(\phi, f[n])$ is denoted by $M(\phi, f)$ whenever it exists.

b) M *infers* the correct answer of $\phi(f)$ if $M(\phi, f) = \phi(f)$.

c) Given $\mathcal{C} \subseteq$ DEC. We write

$$\mathcal{C} \subseteq \text{QAN}\,(L)(M) \text{ if } (\forall \phi \in L)(\forall f \in \mathcal{C})[M(\phi, f) = \phi(f)] \quad \text{and}$$

$$\text{QAN}\,(L) = \{\mathcal{C} \subseteq \text{DEC} : (\exists M)[\mathcal{C} \subseteq \text{QAN}\,(L)(M)]\}.$$

d) The inference types $\text{Q}_i\text{AN}_j\,(L)$ $(i, j \geq 0, i \neq 0)$ denotes the case when the queries considered have at most $i - 1$ alternations of quantifiers and the answer inference machines can make at most j mindchanges. We omit the appropriate subscripts when no restrictions are made in the corresponding case. We will, in text, refer to this as $i = *$ or $j = *$. For example we may say "when $i \in \mathbb{N} \cup \{*\}$ then …". We call $\text{Q}_i\text{AN}_j\,(L)$ *trivial* when DEC $\in \text{Q}_i\text{AN}_j\,(L)$.

The inclusion relations described in Figure 3 follows immediately from Definition 2.4.

$$\begin{array}{ccccccc}
\mathrm{Q}_1\mathrm{AN}\,(L) & \supseteq & \mathrm{Q}_2\mathrm{AN}\,(L) & \supseteq & \cdots & \supseteq & \mathrm{QAN}\,(L) \\
\vdots & & \vdots & & & & \vdots \\
\cup| & & \cup| & & & & \cup| \\
\mathrm{Q}_1\mathrm{AN}_j\,(L) & \supseteq & \mathrm{Q}_2\mathrm{AN}_j\,(L) & \supseteq & \cdots & \supseteq & \mathrm{QAN}_j\,(L) \\
\cup| & & \cup| & & & & \cup| \\
\vdots & & \vdots & & & & \vdots \\
\cup| & & \cup| & & & & \cup| \\
\mathrm{Q}_1\mathrm{AN}_1\,(L) & \supseteq & \mathrm{Q}_2\mathrm{AN}_1\,(L) & \supseteq & \cdots & \supseteq & \mathrm{QAN}_1\,(L) \\
\cup| & & \cup| & & & & \cup| \\
\mathrm{Q}_1\mathrm{AN}_0\,(L) & \supseteq & \mathrm{Q}_2\mathrm{AN}_0\,(L) & \supseteq & \cdots & \supseteq & \mathrm{QAN}_0\,(L)
\end{array}$$

Fig. 3. Hierarchy of Answer Inference Types

Definition 2.5 We call each row (i.e. for $j \in \mathsf{N} \cup \{*\}$) in Figure 3

$$\mathrm{Q}_1\mathrm{AN}_j\,(L) \supseteq\ \mathrm{Q}_2\mathrm{AN}_j\,(L) \supseteq\ \cdots\ \supseteq\ \mathrm{QAN}_j\,(L)$$

a *quantifier* hierarchy of answer inference types and each column (i.e. for $i \in \mathsf{N} \cup \{*\}$) in Figure 3

$$\mathrm{Q}_i\mathrm{AN}_0\,(L) \subseteq\ \mathrm{Q}_i\mathrm{AN}_1\,(L) \subseteq\ \cdots\ \subseteq\ \mathrm{Q}_i\mathrm{AN}\,(L)$$

a *mindchange* hierarchy of answer inference types.

For a general answer inference types $\mathrm{Q}_i\mathrm{AN}_j\,(L)$, the query language used (i.e. L) is also a parameter. In this article we will use the reduction techniques introduced in [17] to deal with this parameter. It was shown that many query languages are reducible to the language $[\mathrm{Succ}, <]^2$. By using reductions, one can obtain results for a class of query languages by simply proving the results for the language $[\mathrm{Succ}, <]^2$. These techniques make use of the decidability results [15,27] about Büchi automata. We will use these tools in Section 7 and Section 8.

2.4 Büchi Automata and $[Succ, <]^2$

We first state some basic facts about Büchi automata and the query language $[\mathrm{Succ}, <]^2$. More specific notions concerning Büchi automata will be introduced in Section 4 and Section 5.

Definition 2.6 ([1,2]) Let Σ be a finite alphabet. A Büchi-automaton

$$\mathcal{A} = (Q, \Sigma, \Delta, s, F)$$

is a *nondeterministic finite automaton* where

a) Q is a (finite) set of states;
b) Σ is a finite alphabet;
c) Δ is a map from $Q \times \Sigma$ to 2^Q;
d) $s \in Q$ and $F \subseteq Q$. s is called the starting state and F is called the set of accepting states.

$\mathcal{A}$ operates on *infinite* strings $\vec{x} \in \Sigma^\omega$. If $\vec{x}(i)$ denotes the $(i+1)^{th}$ symbol of $\vec{x}$, then a *run* of $\mathcal{A}$ on $\vec{x}$ is a sequence of states $\vec{q}$ such that

$$\vec{q}(0) = s \quad \text{and} \quad (\forall i)[\vec{q}(i+1) \in \Delta(\vec{q}(i), \vec{x}(i))].$$

$\mathcal{A}$ *accepts* $\vec{x}$ if there is a run $\vec{q}$ such that $(\exists^\infty i)[\vec{q}(i) \in F]$. $\mathcal{A}$ accepts $\mathcal{C}$ ($\subseteq \Sigma^\omega$) if it accepts precisely the strings in $\mathcal{C}$. A subset $\mathcal{C}$ of Σ^ω is said to be an *ω-regular* language if there is a Büchi-automaton that accepts it. We will also use the term ω automata for Büchi automata.

We use the following connection between the queries in $[\mathrm{Succ}, <]^2$ and Büchi automata.

Theorem 2.7 ([1,2,10]) If $\phi(x_1, \ldots, x_{k_1}, X_1, \ldots, X_{k_2})$ is a formula over the language $[\mathrm{Succ}, <]^2$ then the set A

$$A = \{(a_1, \ldots, a_{k_1}, A_1, \ldots, A_{k_2}) : \phi(a_1, \ldots, a_{k_1}, A_1, \ldots, A_{k_2})\}$$

is ω-regular (here we identify each of the members of A to an ω word via standard coding methods). Furthermore, there is an effective procedure to transform any formula into an appropriate Büchi automaton. (There is also a effective procedure to transform an automaton into an equivalent formula.)

Using this theorem, one can assert that $\phi(X)$ (a query about a set X) can be formulated in $[\mathrm{Succ}, <]^2$ by constructing a suitable Büchi automata.

Example 2.8 Let $\mathcal{A} = (\{s,t,u\}, \{0,1\}, \Delta, \{s\}, \{u\})$ be the Büchi automata where Δ consists of the following rules:

$$\Delta(s,0) = \{s,u\}, \Delta(s,1) = \Delta(t,0) = \{t\};$$

$$\Delta(t,1) = \{s\}, \Delta(u,0) = \{u\} \text{ and } \Delta(u,1) = \emptyset.$$

Then $\mathcal{A}$ accepts exactly the ω strings which have an even number of 1's. By Theorem 2.7 it implies that one can effectively find a query $\psi \in [\mathrm{Succ}, <]^2$ such that $\psi(X)$ is true iff X is a finite set with an even number of elements.

2.5 Reductions

In this paper we use a reduction among query languages [17].

Definition 2.9 Let L_1 and L_2 be two query languages and E be an infinite computable subset of N. Let $f : E \to \mathsf{N}$ be a computable bijection. L_1 is *reducible* to L_2 via the pair (E, f), written as $L_1 \leq_{\mathsf{N}} L_2$ if there is a computable function which satisfies the following two conditions:

a) Domain Condition: The input is a query $\psi_1(X)$ over L_1 and the output is a query $\psi_2(X)$ over L_2.
b) equivalence Condition: $(\forall A \subseteq E)$ $[\psi_1(A) \Leftrightarrow \psi_2\ (f(A))]$.

Note 2.10 In the above formulation, we only ask questions (in L_1) about sets that are subsets of a specific infinite computable set E. We will denote the collection $\mathrm{DEC}_{0,1} \cap 2^E$ by $\mathrm{DEC}_{0,1}(E)$.

Note 2.11 We use the notation $\leq_{\mathsf{N}}$ since it was used in [17]. In that paper we also used variants where N was replaced with other computable sets. We will not consider those cases here.

Fact 2.12 [17]

a) $[+, <] \leq_{\mathsf{N}} [\mathrm{Succ}, <]^2$ where $E = \mathrm{POW}_2$ and f is the function that maps 2^i to i.
b) $[+, <, \mathrm{POW}_b] \leq_{\mathsf{N}} [\mathrm{Succ}, <]^2$ for any $(b \geq 2)$ where $E = \mathrm{POW}_b$ and f is the function that maps b^i to i.

The reduction $L_1 \leq_{\mathsf{N}} L_2$ between the languages L_1 and L_2 allows us to relate concept classes that can be learned via L_1 and L_2. The following technical lemma illustrates this point.

Lemma 2.13 [17] Let $L_1 \leq_{\mathsf{N}} L_2$ via the pair (E, f). Let $\mathcal{I}$ be a passive inference type and $\mathrm{Q}\mathcal{I}$ be the corresponding query notion. Then $\forall \mathcal{C} \subseteq \mathrm{DEC} \cap \{X : X \subseteq E\}$,

a) $\mathcal{C} \in \mathcal{I} \Leftrightarrow f(\mathcal{C}) \in \mathcal{I}$.
b) $\mathcal{C} \in \mathrm{Q}\mathcal{I}\,(L_2) \Rightarrow f^{-1}(\mathcal{C}) \in \mathrm{Q}\mathcal{I}\,(L_1)$.
c) $\mathrm{DEC} \in \mathrm{Q}\mathcal{I}\,(L_1) \Rightarrow \mathrm{DEC} \in \mathrm{Q}\mathcal{I}\,(L_2)$.

Similarly, we can establish the following technical lemma in the context of answer inference.

Lemma 2.14 Let L_1 and L_2 be two reasonable query languages and $\mathcal{I}$ be an inference type. Suppose $L_1 \leq_{\mathsf{N}} L_2$ via the pair $(E,\ f)$ and $\mathcal{C} \subseteq \mathrm{DEC}_{0,1}$. Then

a) $\mathcal{C} \in \mathrm{Q}_i\mathrm{AN}_j\,(L_2) \Rightarrow f^{-1}(\mathcal{C}) \in \mathrm{Q}_i\mathrm{AN}_j\,(L_1)$.
b) $\mathcal{C} \in \mathrm{Q}_i\mathrm{AN}_j\,(L_2) - \mathcal{I} \Rightarrow f^{-1}(\mathcal{C}) \in \mathrm{Q}_i\mathrm{AN}_j\,(L_1) - \mathcal{I}$.

Proof:

a) Suppose that $L_1 \leq_{\mathsf{N}} L_2$ via the pair (E,f). Let $\mathcal{C}$ be a collection of sets such that $\mathcal{C} \in \mathrm{Q}_i\mathrm{AN}_j\,(L_2)$. By the property of reduction, $f^{-1}(\mathcal{C}) \in \mathrm{Q}_i\mathrm{AN}_j\,(L_1)$.
b) Suppose that $\mathcal{C} \in \mathrm{Q}_i\mathrm{AN}_j\,(L_2) - \mathcal{I}$. By previous part of this lemma, $f^{-1}(\mathcal{C}) \in \mathrm{Q}_i\mathrm{AN}_j\,(L_1)$. By Lemma 2.13, if $f^{-1}(\mathcal{C}) \in \mathcal{I}$ and as f is a computable *bijection*, $\mathcal{C} = f(f^{-1}(\mathcal{C})) \in \mathcal{I}$, which is a contradiction. Hence $f^{-1}(\mathcal{C}) \notin \mathcal{I}$.

▮

Note 2.15 In [17], some query languages that are extensions of $[\mathrm{Succ}, <]^2$ via an additional predicate P, are studied. They are shown to be reducible to $[\mathrm{Succ}, <]^2$ via $\leq_w$, a weaker form of reduction. We do not use this weaker form of reduction and hence we do not include it.

3 Technical Summary

For simplicity, we will adopt the following notations in this summary. We use

- L_s to denote any query languages that are reducible to $[\mathrm{Succ}, <]^2$ (i.e. $L \leq_{\mathsf{N}} [\mathrm{Succ}, <]^2$).
- L_d to denote any reasonable query languages that have decidable base languages.
- $L_<$ to denote query languages that can express the relation $<$.

3.1 When is an Answer Inference Type Trivial?

1) Existential query about any computable function formulated in any first order query language L_d can be answered with at most one mindchange and this bound is tight. This result also holds for any second order languages L_P, where $L_P = [\mathrm{Succ}, <]^2$, $[\mathrm{Succ}, <, \mathrm{FAC}]^2$ or

$$L_P \in \{[\mathrm{Succ}, <, \mathrm{POW}_b]^2, [\mathrm{Succ}, <, \mathrm{POLY}_b]^2 : b > 1\}.$$

Hence the corresponding answer inference types are trivial.

2) More complex queries formulated in L_d (those with at least one alternation of quantifiers) about computable functions cannot be inferred.

3.2 Separating the Mindchange Hierarchies

For existential queries: the mindchange hierarchy collapses to the *first* level for any L_d. For more complex queries: the mindchange hierarchies are strict for any L_s.

3.3 Separating the Quantifier Hierarchies

The quantifier hierarchies for two particular query languages collapse when we restrict input to computable sets. In short, we have

$$\mathrm{QAN}\,([\mathrm{Succ},<]^2) = \mathrm{Q_2AN}\,([\mathrm{Succ},<]^2)$$

$$\mathrm{QAN}\,([\mathrm{Succ},<]) = \mathrm{Q_3AN}\,([\mathrm{Succ},<])$$

3.4 Inferring Answers Versus Inferring Programs

- There are situations in which inferring programs via some restrictive criterion is easy but inferring answers is hard. Formally,

$$\mathrm{PEX}_0 \not\subseteq \mathrm{Q_2AN}\,(L_d).$$

- There are situations in which inferring programs via some generous criterion is hard but inferring answers is easy. Formally,

$$(\forall a, n, n \geq 1)[\mathrm{QAN}\,(L_s) \not\subseteq [1,n]\mathrm{QBC}^a\,(L_s)].$$

Note that $\mathrm{QAN}_0\,(L_s)$ (resp. $\mathrm{Q_2AN}\,(L_s)$) is the smallest (resp. largest) non-trivial answer inference types. These results show that the learning of programs and the learning of properties are incomparable in a strong sense.

3.5 Lifting Results to Query Inference Types

In *query inference* the learner is trying to learn a function and is allowed to ask questions in some language (e.g., $L = [+,<]$) about it (e.g., $(\exists x)(\forall y)[(x < y) \rightarrow \mathcal{F}(x) = 0]$). All of the standard inference types (e.g. EX) have query analogs (e.g., QEX[<]). This notion has been studied extensively in [17–19]; we will review the basic definitions and results in Section 8.1.

It was conjectured that if two inference types differed then their query analogs differed. Using the machinery in this paper we show that, for many inference types $\mathcal{I}$ and $\mathcal{J}$ (which includes EX, BC and their variants with respect to mindchanges and teams), for many languages L,

$$\mathcal{J} - \mathcal{I} \neq \emptyset \ \Rightarrow \ \mathrm{Q}\mathcal{J}\,(L) - \mathrm{Q}\mathcal{I}\,(L) \neq \emptyset.$$

We show that:

1) For $i \geq 1$, $[1,i]\mathrm{QEX}\,(L_s) \subset [1,i+1]\mathrm{QEX}\,(L_s)$.
2) For $i \geq 1$, $[1,i]\mathrm{QBC}\,(L_s) \subset [1,i+1]\mathrm{QBC}\,(L_s)$.
3) For c,d such that $24/49 < c/d$

$$[24,49]\mathrm{QEX}_0\,(L_s) - [c,d]\mathrm{QEX}\,(L_s) \neq \emptyset.$$

4) $$[1,2]\mathrm{QEX}_0\,(L_s) \subset [2,4]\mathrm{QEX}_0\,(L_s).$$

4 When is an Answer Inference Type Trivial?

Let $i > 0$ or $i = *$, and $j \geq 0$ or $j = *$. In this section, we will show that for these values of i and j, an answer inference type $\mathrm{Q}_i\mathrm{AN}_j\,(L)$ is trivial iff $i = 1$ and $j \neq 0$.

Theorem 4.1 DEC $\notin \mathrm{Q}_1\mathrm{AN}_0\,([\emptyset])$. Hence, DEC $\notin \mathrm{Q}_1\mathrm{AN}_0\,(L)$ for any query languages L.

Proof: Let $S = \{A \subseteq \mathsf{N} : |A| \leq 1\}$ and ϕ be the query $\phi(X) = (\exists y)[y \in X]$. Here we represent members of S by their characteristic strings. That is, members of S are either 0^ω or 0^i10^ω for some $i \geq 0$. Suppose that $S \subseteq \mathrm{Q}_1\mathrm{AN}_0\,(M)$ for some AIM M. When we feed M with initial segments of the form 0^i ($i \geq 0$, i increasing), then there is a k such that M output an answer b after examining the segment 0^k. However, the sets $A, B \in S$ where $A = 0^\omega$ and $B = 0^k10^\omega$ are both extensions of σ_k. If $b = 1$ (resp $b = 0$) then $\phi(A) = 1 - b$ (resp. $\phi(B) = 1 - b$), which is a contradiction. Therefore DEC $\notin \mathrm{Q}_1\mathrm{AN}_0\,([\emptyset])$ and hence DEC $\notin \mathrm{Q}_1\mathrm{AN}_0\,(L)$ for any reasonable query languages L. ▮

When mindchanges are allowed, we show that existential queries are learnable with one mindchange. We need the technical lemmas from Section 3 of [19]. We state them using the terminology in this paper.

Lemma 4.2 ([19]) Let L be a *first order* query language with a decidable base language. Let ϕ be an existential query in L. Then

1) There is an effective procedure which will convert ϕ to an equivalent query ϕ' where ϕ' is of the form

$$\phi' = (\exists \vec{z})[\theta(\vec{z}, \mathcal{F}(t_1), \mathcal{F}(t_2), \ldots, \mathcal{F}(t_m))],$$

where ϕ' does not have any nested occurrences of the function symbol $\mathcal{F}$ ($\mathcal{F}$ does not appear in any of the t_i's). In addition, each t_i is a term that depends on the variables $\vec{z}$, and θ is in prenex normal form. For each t_i we may write it as $t_i(\vec{z})$.

2) For any $f \in$ DEC the query $\phi(f)$ is true iff there *exists* an assignment $\vec{z} = \vec{c}$ ($\vec{c}$ is a vector of natural numbers) such that the *sentence*

$$\theta(\vec{c}, f(t_1(\vec{c})), f(t_2(\vec{c})), \ldots, f(t_m(\vec{c})))$$

is true.

Proof: We refer the interested readers to Section 3 in [19] for a formal presentation of these results. ∎

Theorem 4.3 Let L be a *first order* query language with a decidable base language. Then DEC $\in \mathrm{Q}_1\mathrm{AN}_1\,(L)$.

Proof: By Lemma 4.2, we may assume that the query is of the form

$$(\exists \vec{z})[\theta(\vec{z}, f(t_1(\vec{z})), f(t_2(\vec{z})), \ldots, f(t_m(\vec{z})))].$$

Initially output NO. Then, as you see $f(0), f(1), f(2), \ldots$ dovetail on both the values of f seen and all $\vec{c}$ to see if $\theta(\vec{c}, f(t_1(\vec{c})), f(t_2(\vec{c})), \ldots, f(t_m(\vec{c})))$ is ever true. If this ever happens output YES. If the query is true then some value that makes it true will appear. That will be one mindchange. If the query is false then it will keep saying NO and make no mindchanges. ∎

Since with only one mindchange we obtain DEC, we have the following corollary. It states that a mindchange hierarchy collapses.

Corollary 4.4 Let L be a *first order* query language with a decidable base language. Then

$$\mathrm{Q}_1\mathrm{AN}_0\,(L) \subset \mathrm{Q}_1\mathrm{AN}_1\,(L) = \cdots = \mathrm{Q}_1\mathrm{AN}_i\,(L) = \cdots = \mathrm{Q}_1\mathrm{AN}\,(L)\ (i \geq 1).$$

We will now show that Theorem 4.3 can be extended to the second order query language $[\mathrm{Succ}, <]^2$, with the caveat that we use $\mathrm{DEC}_{0,1}$ instead of DEC.

Theorem 4.5 $\mathrm{DEC}_{0,1} \in \mathrm{Q}_1\mathrm{AN}_1\,([\mathrm{Succ}, <]^2)$.

Proof: Let ψ be an existential query in $[\mathrm{Succ}, <]^2$. By using the same procedure as in Lemma 4.2.1, we may assume that ψ has no nesting of the function symbol $\mathcal{F}$. It follows from the definition of terms in $[\mathrm{Succ}, <]^2$ (see Definition 2.3) that each terms in ψ only depends on first order variables. We may suppose that ψ is of the following form:

$$\psi = (\exists \vec{X})(\exists \vec{z})[\theta(\vec{X}, \vec{z}, \mathcal{F}(t_1(\vec{z})), \mathcal{F}(t_2(\vec{z})), \ldots, \mathcal{F}(t_m(\vec{z})))] \tag{1}$$

The following procedure will learn the answers of $\psi(f)$ with at most 1 mind-change.

Procedure for learning $\psi(f)$

By previous argument, we may assume that ψ is of the form as in (1).

a) Output the answer FALSE.

b) Dovetail w.r.t. the assignment $\vec{z} = \vec{c}$. For each of assignment $\vec{z} = \vec{c}$, perform the following steps:

(1) compute the terms $t_1(\vec{c}), \ldots, t_m(\vec{c})$;

(2) read in an initial segment of f so the terms $\mathcal{F}(t_1(\vec{c})), \ldots, \mathcal{F}(t_m(\vec{c}))$ can be interpreted w.r.t. the function f. That is, all the values of $f(t_1(\vec{c}))$, …, $f(t_m(\vec{c}))$ are obtained.

(3) When we reach this step, we may assume that the query is of the form

$$(\exists \vec{X})[\theta'(\vec{X})].$$

At this stage θ' is in prenex normal form and all the terms have already been evaluated. Atomic formulas are either of the form

- $(c_1 = c_2)$ or $(c_1 < c_2)$ where c_1 and c_2 are two natural numbers. The truth of these atomic formulas can be determined easily.
- $(c \in X)$, where c is a natural number and X is a set variable. The existence of sets that can satisfy a boolean combination of these *constraints* can also be determined effectively.

Let $\vec{X} = (X_1, \ldots, X_m)$. To determine the truth of $(\exists \vec{X})[\theta'(\vec{X})]$, it is equivalent to determine the existence of (possibly infinite) sets $X_1, \ldots, X_m$ that satisfy the collection of constraints described by the atomic formulas in θ'. Note that we are only putting a finite number of conditions on the sets and the conditions involve concrete natural numbers. Hence there exists (possibly infinite) sets $X_1, \ldots, X_m$ that satisfy the collection of constraints described by the atomic formulas in θ' iff there exists finite sets $X_1, \ldots, X_m$ that satisfy the collection of constraints described by the atomic formulas in θ'. This can be easily determined.

From the above observation, this problem is clearly decidable. If the truth value is True, then output the truth value and terminate the program. Otherwise, go back to the dovetailing step.

End of Procedure

It is easy to see that the procedure can determine the truth of an existential query with at most 1 mindchange. ▮

The proof of Theorem 4.5 can be modified to obtain the same result with any second order query language with decidable base. Hence we have the following Porism[3]

Porism 4.6 Let L be one the following query languages: $[\mathrm{Succ}, <]^2$,$[\mathrm{Succ}, <, \mathrm{POW}_b]^2$, $[\mathrm{Succ}, <, \mathrm{POLY}_b]^2$ and $[\mathrm{Succ}, <, \mathrm{FAC}]^2$. Then

$$\mathrm{DEC} \in \mathrm{Q}_1\mathrm{AN}_1\,(L).$$

Hence for existential queries, the mindchange hierarchy collapses in the following way:

Corollary 4.7 Let $b > 1$. Let L be either a first order query language or one the following query languages: $[\mathrm{Succ}, <]^2$,$[\mathrm{Succ}, <, \mathrm{POW}_b]^2$, $[\mathrm{Succ}, <, \mathrm{POLY}_b]^2$ and $[\mathrm{Succ}, <, \mathrm{FAC}]^2$. Then

$$\mathrm{Q}_1\mathrm{AN}_0\,(L) \subset \mathrm{Q}_1\mathrm{AN}_1\,(L) = \cdots = \mathrm{Q}_1\mathrm{AN}_i\,(L) = \cdots = \mathrm{Q}_1\mathrm{AN}\,(L)\ (i \geq 1).$$

Theorem 4.3 can also be generalized to the following case.

Corollary 4.8 Let L be a first order query language. Then $\mathrm{Q}_i\mathrm{AN}_0\,(L) \subseteq \mathrm{Q}_{i+1}\mathrm{AN}_1\,(L)$ for all $i \geq 1$.

Proof: Let $\mathcal{C} \subseteq \mathrm{DEC}$ such that $\mathcal{C} \in \mathrm{Q}_i\mathrm{AN}_0\,(L)$, we show that $\mathcal{C} \subseteq \mathrm{Q}_{i+1}\mathrm{AN}_1\,(L)$. It suffices to look at those queries that are of the form

$$(Q_1\vec{x}_1)(Q_2\vec{x}_2)\ldots(Q_{k+1}\vec{x}_{k+1})\alpha(\mathcal{F}, \vec{x}_1, \vec{x}_2, \ldots, \vec{x}_{k+1})$$

where $Q_1, Q_2, \ldots, Q_{k+1}$ are either $\exists$ or $\forall$ and they are alternating. Without loss of generality we may assume that $Q_1 = \exists$. By dovetailing one may effectively

[3] A porism is a statement that follows from a theorem by using the same proof technique. Note that a Corollary can be derived from the Theorem, which is different.

list all possible substitutions to the variable vector $\vec{x}_1$. By assumption there is an AIM M such that for any $f \in \mathcal{C}$, the queries

$$(Q_2\vec{x}_2)\dots(Q_{k+1}\vec{x}_{k+1})\alpha(f, \vec{c}, \vec{x}_2, \dots, \vec{x}_{k+1})$$

($\vec{c}$ ranges over all possible substitutions of $\vec{x}_1$) can be *decided* by M. By a similar argument as in the proof of Theorem 4.3 we get $\mathcal{C} \in \mathrm{Q}_{k+1}\mathrm{AN}_1\,(L)$. ▮

Note 4.9 By Theorem 5.4 and Corollary 5.5 (which will be shown in the next section), we can show that the subset relation is actually *proper* for many query languages. For any $i \geq 2$, we have

$$L \leq_{\mathsf{N}} [\mathrm{Succ}, <]^2 \Rightarrow \mathrm{Q}_i\mathrm{AN}_0\,(L) \subset \mathrm{Q}_{i+1}\mathrm{AN}_1\,(L).$$

The case when $i = 1$ can also by showed easily by considering the class of sets that are either a singleton set or the empty set.

While existential queries about computable functions are easy to infer, more complex properties of computable functions cannot be inferred.

Theorem 4.10 DEC $\notin \mathrm{Q}_2\mathrm{AN}\,(\emptyset)$.

Proof: Suppose DEC $\subseteq \mathrm{Q}_2\mathrm{AN}\,([\emptyset])(M)$. We give a query ϕ in the language $[\emptyset]$ and construct a computable function f (using ϕ and M) such that if $b = M(\phi, f)$, then $\phi(f) = 1 - b$. Let $\phi = (\forall x)(\exists y)[\mathcal{F}(y) = x]$. ϕ is true iff f is surjective. By assumption, we may assume that $M(\phi, \sigma)$ converges for any possible initial segment σ. We construct a computable function f as follows. We set $f(0) = 0$. Assume we have constructed a finite segment $f[n]$ of f. Let b=$M(\phi, f[n])$. If $b = 0$, then set $f(n+1) = f(n) + 1$. Otherwise set $f(n+1) = f(n)$. It is easy to see either $M(\phi, f)$ does not converge or $M(\phi, f)$ is not the right answer. ▮

In later sections we shall examine the *nontrivial* answer inference types.

5 Separating the Mindchange Hierarchies

Corollary 4.7 shows that when dealing with existential questions, the corresponding mindchange hierarchy collapses drastically. By contrast, we show that when $L \leq_{\mathsf{N}} [\mathrm{Succ}, <]^2$ and when there is at least one alternation of quantifiers, all the inclusions in the corresponding mindchange hierarchies are *proper*, even when we restrict the input to computable *sets*.

Definition 5.1 Let $A \subseteq \mathsf{N}$ and L be a reasonable query language. A is said to be *definable by a query* ϕ in L if there is a query $\phi(X) \in L$ such that $\phi(X)$ is true if and only if $X = A$.

Intuitively, these sets are those that can be *described* precisely by a query. For example, any finite and co-finite sets are definable via a query in any reasonable query language. The set of all even numbers is also definable in the language [Succ] via the query

$$\phi(X) = (0 \in X) \wedge (1 \notin X) \wedge (\forall y)[y \in X \Leftrightarrow \text{ Succ}(\text{Succ}(y)) \in X]$$

We will now show that the mindchange hierarchies for non-existential queries are proper. To simplify our presentation, we introduce the following definitions.

Definition 5.2

a) Let $A, B \subseteq \mathsf{N}$. B is said to be a k-variant of A if the *symmetric difference* of the sets A and B (denoted by $A\Delta B$) has precisely k elements. Note that the size of $A\Delta B$ can be written as $|A - B| + |B - A|$.

b) For any $A \in \text{DEC}_{0,1}$, we define the *k-ball* of A to be the set

$$\mathcal{S}_k(A) = \{B \in \text{DEC}_{0,1} : |A\Delta B| \leq k\}.$$

Lemma 5.3

a) Let A be a set that is definable by a query ϕ_A in the language L. Then for any $j, m \in \mathsf{N}$, the question

'Is X a j-variant of A, where j is an odd number and $j \leq m$?'

can be expressed as a query in L of the form

$$(\exists Y)(\exists x_1, \ldots, x_k)(\forall y)[E_1 \wedge E_2 \wedge E_3].$$

We will denote this query by $\eta_{\text{odd}}(A, X, m)$.

b) For any set A that is definable by a query ϕ_A in the language $[\text{Succ}, <]^2$ and for any $k \geq 1$,

$$\mathcal{S}_k(A) \in \text{QAN}_k\,([\text{Succ}, <]^2).$$

Proof:

a) First note that

$$u \in X\Delta Y \quad \text{iff} \quad [[(u \in X) \wedge (u \notin Y)] \vee [(u \notin X) \wedge (u \in Y)]]$$

and that A is definable in L via ϕ_A. The question

'Is X a j-variant of A ?'

can be represented by the following query $\phi_{j,A}(X)$ in L:

$$\phi_{j,A}(X) = (\exists Y)(\exists x_1)(\exists x_2)\cdots(\exists x_j)(\forall y)(E_1 \wedge E_2 \wedge E_3)$$

where E_1, E_2 and E_3 are the expressions

$$E_1 : \bigwedge_{n\neq m, n,m\leq j} (x_n \neq x_m); \quad E_2 : \phi_A(Y); \quad E_3 : [(y \in Y\Delta X) \Leftrightarrow \bigvee_{n=1}^{j} (y = x_n)]$$

Hence, the question

‘Is X a j-variant of A, where j is an odd number and $j \leq m$?’

can be expressed as

$$\eta_{\text{odd}}(A, X, m) = \bigvee_{j\in\{1,\ldots,m\}\wedge j \text{ odd}} \phi_{j,A}(X).$$

As each $\phi_{j,A}(X)$ is a query in L, $\eta_{\text{odd}}(A, X, m)$ is a query in L.

b) Let $\phi(X) \in [\text{Succ}, <]^2$ and $B \in \mathcal{S}_k(A)$, we can infer the answer to $\phi(B)$ as follows. Initially we guess that $\phi(B)$=$\phi(A)$. To decide the truth of $\phi(A)$, we construct a Büchi automaton $\mathcal{A}_\phi$ for $\phi(X)$ (Keep this automaton, we will use it later in the algorithm also.) Since A is definable by a query in $[\text{Succ}, <]^2$, we can construct a Büchi automaton $\mathcal{A}$ that accepts only A. Now construct the intersection of $\mathcal{A}_\phi$ and $\mathcal{A}$ and test for emptiness. As the emptiness problem of Büchi automata is decidable, the truth of the query $\phi(A)$ can be obtained. Next, on input $\sigma \preceq B$, $|\sigma| = n$, we first find a Büchi automaton $\mathcal{A}_\sigma$ that accepts the string $\sigma A(n)A(n+1)\cdots$. We then construct the intersection of $\mathcal{A}_\phi$ and $\mathcal{A}_\sigma$ and test for emptiness. Note that over time the set we are testing for intersection will change at most k times since B is in the k-ball of A.

▮

Theorem 5.4 Let $i, k \in \mathsf{N} \cup \{*\}, i \neq 0, 1$ and $k \neq 0, *$.

a) $\text{QAN}_k\,([\text{Succ}, <]^2) \not\subseteq \text{Q}_2\text{AN}_{k-1}\,([\text{Succ}, <]^2)$.

b) $$\text{Q}_i\text{AN}_0\,([\text{Succ}, <]^2) \subset \text{Q}_i\text{AN}_1\,([\text{Succ}, <]^2) \subset \cdots$$
$$\cdots \subset \text{Q}_i\text{AN}_j\,([\text{Succ}, <]^2) \subset \cdots \subset \text{Q}_i\text{AN}\,([\text{Succ}, <]^2).$$

Proof:

a) From part *b)* of Lemma 5.3 it suffices to show that for any set A that is definable via a query in $[\text{Succ}, <]^2$ and for any $k \geq 1$,

$$\mathcal{S}_k(A) \notin \text{Q}_2\text{AN}_{k-1}\,([\text{Succ}, <]^2).$$

Assume by way of contradiction that $\mathcal{S}_k(A) \subseteq \mathrm{Q}_2\mathrm{AN}_{k-1}([\mathrm{Succ}, <]^2)(M)$ for some positive integer k. Let ϕ be the query $\eta_{\mathrm{odd}}(A, X, k)$ (from Lemma 5.3). Since A is definable via a query in $[\mathrm{Succ}, <]^2$, by part a) of Lemma 5.3, $\phi = \eta_{\mathrm{odd}}(A, X, k)$ is a query in $[\mathrm{Succ}, <]^2$.

We shall construct a computable set $B \in \mathcal{S}_k(A)$ by diagonalizing against M using ϕ as our input. B will witness the fact that M cannot infer the correct answer to the query $\phi(B)$. Throughout the construction, at the beginning of any stage s we keep track of the following:

(1) B_s, the initial segment of the set B defined so far.
(2) A_s, the initial segment of A such that dom(A_s)=dom(B_s).
(3) d_s=$|\{x : x \in A\Delta B, x \in \mathrm{dom}(A_s)\}|$.
(4) c_s, the number of mindchanges the machine has made at the beginning of that stage.

Construction of B

Let A_0 be the initial segment of A where dom(A_0)=$\{0\}$. Let B_0=A_0; $a_0 = 0$; $c_0 = 0$.

Stage s: Compute $b = M(\phi, B_s)$ and d_s. Compute a_{s+1}, which is the smallest number in $A - \{0, \ldots, a_s\}$. Let A_{s+1} be the initial segment of A with domain $\{0, \ldots, a_{s+1}\}$. Define

$$C = \{(u, 0) : a_s < u < a_{s+1}\}.$$

Define B_{s+1} as follows.

(1) If $b = TRUE$ and d_s is odd then the question "is $|B_s\Delta A|$ odd" is currently true, and is being answered correctly. We want to make it answered incorrectly. Hence let $B_{s+1} = B_s \cup C \cup \{(a_{s+1}, 0)\}$. This creates another place where A and B differ, so they now differ in an even number of places.
(2) If $b = FALSE$ and d_s is odd then the question "is $|B_s\Delta A|$ odd" is currently false, but is being answered incorrectly. We want to maintain this. $B_s \cup C \cup \{(a_{s+1}, 1)\}$. This does not create another place where they differ, so they still differ in an odd number of places.
(3) If $b = TRUE$ and d_s is even then the question "is $|B_s\Delta A|$ odd" is currently false, but is being answered incorrectly. Hence let $B_{s+1} = B_s \cup C \cup \{(a_{s+1}, 1)\}$.
(4) If $b = TRUE$ and d_s is odd then the question "is $|B_s\Delta A|$ odd" is currently true, but is being answered correctly. Hence let $B_{s+1} = B_s \cup C \cup \{(a_{s+1}, 0)\}$.

Compute $M(\phi, B_{s+1})$. If this is not equal to b then set $c_{s+1} = c_s + 1$.

End of Construction

Let c be the number of times that M changes its mind during the construction. By definition of M, $c < k$, hence there is a stage we denote t such that at any stage t' with $t' > t$, $M(\phi, B_{t'}) = M(\phi, B_t) = M(\phi, B)$. During stage t of the construction, either there was a mindchange or there was not. If there was no mindchange then the disagreement that we caused in this stage is permanent, so $1-M(\phi,B_t){=}1-M(\phi,B)$ and this answer is wrong. If there was a mindchange then during the next stage, which we call t', we will cause a disagreement that will be permanent. Hence $1-M(\phi,B_{t'}){=}1-M(\phi,B)$ and this answer is wrong.

Finally, we note that by choosing $A=\mathsf{N}$, the query $\eta_{\mathrm{odd}}(A,X,j)$ $(j\geq 1)$ is a $\exists\forall$ query in the language $[\emptyset]$. Therefore,

$$\mathcal{S}_k(\mathsf{N})\in \mathrm{QAN}_k\left([\mathrm{Succ},<]^2\right)-\mathrm{QAN}_k\left([\emptyset]\right)$$

and the result follows immediately.

b) By Theorem 6.1 for any $i\geq 2$ and $k\geq 1$,

$$\mathrm{QAN}_k\left([\mathrm{Succ},<]^2\right)\subseteq \mathrm{Q}_i\mathrm{AN}_k\left([\mathrm{Succ},<]^2\right) \text{ and}$$

$$\mathrm{Q}_i\mathrm{AN}_{k-1}\left([\mathrm{Succ},<]^2\right)\subseteq \mathrm{Q}_2\mathrm{AN}_{k-1}\left([\mathrm{Succ},<]^2\right).$$

Suppose that $\mathrm{Q}_i\mathrm{AN}_k\left([\mathrm{Succ},<]^2\right)=\mathrm{Q}_i\mathrm{AN}_{k-1}\left([\mathrm{Succ},<]^2\right)$. Then

$$\begin{array}{rcl}\mathrm{QAN}_k\left([\mathrm{Succ},<]^2\right) & \subseteq & \mathrm{Q}_i\mathrm{AN}_k\left([\mathrm{Succ},<]^2\right)\\ & = & \mathrm{Q}_i\mathrm{AN}_{k-1}\left([\mathrm{Succ},<]^2\right)\\ & \subseteq & \mathrm{Q}_2\mathrm{AN}_{k-1}\left([\mathrm{Succ},<]^2\right),\end{array}$$

which implies $\mathrm{QAN}_k\left([\mathrm{Succ},<]^2\right)\subseteq \mathrm{Q}_2\mathrm{AN}_{k-1}\left([\mathrm{Succ},<]^2\right)$. This contradicts part (*a*). Thus, for any $i\geq 2$ and $k\in\mathsf{N}$,

$$\mathrm{Q}_i\mathrm{AN}_k\left([\mathrm{Succ},<]^2\right)\subset \mathrm{Q}_i\mathrm{AN}_{k+1}\left([\mathrm{Succ},<]^2\right).$$

Finally, suppose $\mathrm{Q}_i\mathrm{AN}\left([\mathrm{Succ},<]^2\right)=\mathrm{Q}_i\mathrm{AN}_n\left([\mathrm{Succ},<]^2\right)$, for some $n\in\mathsf{N}$. We then have

$$\begin{array}{rcl}\mathrm{Q}_i\mathrm{AN}\left([\mathrm{Succ},<]^2\right) & = & \mathrm{Q}_i\mathrm{AN}_n\left([\mathrm{Succ},<]^2\right)\\ & \subset & \mathrm{Q}_i\mathrm{AN}_{n+1}\left([\mathrm{Succ},<]^2\right)\\ & = & \mathrm{Q}_i\mathrm{AN}\left([\mathrm{Succ},<]^2\right),\end{array}$$

a contradiction. Therefore, for $i\neq 0,1$ and $k\in\mathsf{N}$,

$$\mathrm{Q}_i\mathrm{AN}_k\left([\mathrm{Succ},<]^2\right)\subset \mathrm{Q}_i\mathrm{AN}\left([\mathrm{Succ},<]^2\right)$$

∎

Corollary 5.5 Let $i,k\in\mathsf{N}\cup\{*\}, i\neq 0,1$ and $k\neq 0$.

a) $(\forall L\leq_{\mathsf{N}}[\mathrm{Succ},<]^2)$ [$\mathrm{QAN}_k(L)\not\subseteq\ \mathrm{Q}_2\mathrm{AN}_{k-1}(L)$].

b) For all b, $\mathrm{QAN}_k\,([+,<,\mathrm{POW}_b]) \not\subseteq \mathrm{Q}_2\mathrm{AN}_{k-1}\,([+,<,\mathrm{POW}_b])$.
c) $(\forall L \leq_{\mathsf{N}} [\mathrm{Succ},<]^2)[\ \mathrm{Q}_i\mathrm{AN}_0\,(L) \subset \mathrm{Q}_i\mathrm{AN}_1\,(L) \subset \cdots \subset \mathrm{Q}_i\mathrm{AN}\,(L)\ ]$.
d) For all b,

$$\mathrm{Q}_i\mathrm{AN}_0\,([+,<,\mathrm{POW}_b]) \subset \mathrm{Q}_i\mathrm{AN}_1\,([+,<,\mathrm{POW}_b]) \subset \cdots$$

$$\cdots \subset \mathrm{Q}_i\mathrm{AN}_j\,([+,<,\mathrm{POW}_b]) \subset \cdots \subset \mathrm{Q}_i\mathrm{AN}\,([+,<,\mathrm{POW}_b]).$$

Proof: By Theorem 5.4, and Fact 2.12. ▮

6 Quantifier Hierarchies

It is natural to ask if the quantifier hierarchies are also strict. In this subsection we will give examples of quantifier hierarchies that collapse to *low* levels.

First, let L be either the language $[\mathrm{Succ},<]$ or $[\mathrm{Succ},<]^2$. Suppose that $\phi(X)$ is a query about a set X in either one of these languages. We claim that these queries cannot be *too* complex. From Theorem 2.7 we see that to test if $\phi(A)$ is true, it is equivalent to test if the characteristic string of A is accepted by a corresponding ω automata $\mathcal{A}_\phi$. This automata can be constructed effectively from ϕ. As the *acceptance condition* of any ω automata is itself a formula in $[\mathrm{Succ},<]^2$ which uses *few* alternations of quantifiers, these queries cannot be too complex. By exploring the quantifier complexity of these acceptance formulas, we have the following results:

Theorem 6.1 When we restrict input to sets, we have

a) $\mathrm{Q}_2\mathrm{AN}\,([\mathrm{Succ},<]^2) = \cdots = \mathrm{Q}_j\mathrm{AN}\,([\mathrm{Succ},<]^2) = \cdots = \mathrm{QAN}\,([\mathrm{Succ},<]^2)$
b) $\mathrm{Q}_3\mathrm{AN}\,([\mathrm{Succ},<]) = \cdots = \mathrm{Q}_j\mathrm{AN}\,([\mathrm{Succ},<]) = \cdots = \mathrm{QAN}\,([\mathrm{Succ},<])$

Proof:

a) By [29], all ω regular languages can be written as the set of solutions to a two quantifier formula in the language in $[\mathrm{Succ},<]^2$.
b) By [37], any query in $[\mathrm{Succ},<]$ is equivalent to a boolean combination of formulas in $[\mathrm{Succ},<]$, where each formula has at most *two alternations* of quantifiers.

▮

Note 6.2 The results from [37] also imply the following results in answer inference and query inference (see Section 5 for basic notions on query inference).

1) When restrict input to sets, QEX $([\text{Succ}, <]^2) = \text{Q}_2\text{EX}\ [[\text{Succ}, <]]^2$.
2) When restrict input to sets, QEX $([\text{Succ}, <]) = \text{Q}_3\text{EX}\ ([\text{Succ}, <])$.

Hence, queries in the languages [Succ, <] and $[\text{Succ}, <]^2$ are not too *expressive.* For $L \leq_{\mathsf{N}} [\text{Succ}, <]^2$ via the pair (E,f), one can use the same idea to show that the corresponding quantifier hierarchy collapses *locally.* That is, the quantifier hierarchy collapses when we restrict our input to computable subsets of E.

It is an open question to determine if other quantifier hierarchies collapse.

7 Inferring Answers versus Inferring Programs

Recall that for many query languages L, the largest nontrivial answer inference type is $\text{Q}_2\text{AN}\,(L)$ and the smallest is $\text{QAN}_0\,(L)$ (see Figure 3). In this section we will use them to demonstrate that the learning of answers and the learning of programs are incomparable in a very strong sense. Our comparisons highlight the following two extreme cases:

First Case: There are concepts that can be learned using a very restrictive identification criteria. However, relatively simple properties of the functions in this class are not learnable.

Second Case: There are concepts that cannot be learned using a very generous identification criteria. However, all boolean queries about the functions in this class are learnable for many query languages.

7.1 First Case

Consider the Popperian inference type PEX_0. The learning criteria is very restrictive. It does not allows mindchanges and any conjectures that the machine can make must be a *total* computable function. We compare it with the largest *nontrivial* inference type $\text{Q}_2\text{AN}\,(L)$. The following non-inclusion holds:

Theorem 7.1 $\text{PEX}_0 \not\subseteq \text{Q}_2\text{AN}\,(\emptyset)$.

Proof: Let $P_0, P_1, P_2, \ldots$ an effective enumeration of the primitive recursive functions that (1) satisfies m-ary composition, (2) satisfies the s-m-n theorem, and (2) function $E(i, x) = P_i(x)$ is computable. (Note that any reasonable encoding of the primitive recursive functions will have these properties.). Consider the concept

$$A = \{f : P_{f(0)} = f\}.$$

Intuitively the value $f(0)$ is the index of a primitive recursive function that is equal to f. Clearly $A \in \mathrm{PEX}_0$.

Assume, by way of contradiction, that $\mathrm{PEX}_0 \subseteq \mathrm{Q}_2\mathrm{AN}\,(L)$ via machine M. We can assume that M is primitive recursive by slowing it down. Let ϕ be the query

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

$\phi(f)$ is true iff f is surjective. We construct a computable function $f \in A$ such that $M(\phi, f) \neq \phi(f)$.

Construction of f

Stage 0: Let $f(0)$ be the index of this construction for f. (We can do this by the recursion theorem for primitive recursive functions. See Theorem 4.6 of [28].) Let

$f(1) = 0,$

$f(2) = 1,$

$\vdots$

$f(f(0))) = f(0) - 1,$

and let

$f(f(0) + 1) = f(0) + 1.$

Let $x_0 = f(0) + 1$. At every state x_s will be such that, at the end of stage s, $f(0), \ldots, f(x_s)$ are defined.

Stage $s + 1$:

Compute $b = M(\phi, f(0)f(1)\cdots f(x_s))$. If $b = 0$ (so $M(\phi, f(0)f(1)\cdots f(x_s))$ thinks that f is not onto) then let $f(x_s + 1) = f(x_s) + 1$. If $b = 1$ (so

$M(\phi, f(0)f(1)\cdots f(x_s))$ thinks that f is onto) then let $f(x_s+1) = f(x_s)$.

End of Construction

By Stage 0, $f \in A$. If $M(\phi, f)$ converges to 0 then f will keep outputting numbers in order and hence f is onto. If $M(\phi, f)$ converges to 1 then f will keep outputting the same number and hence f is not onto. Hence if $M(\phi, f)$ converges then it is incorrect. This is a contradiction. ▮

7.2 The Omega Operator Ω

We consider another extreme case where we will show that the smallest non-trivial inference type $\mathrm{QAN}_0(L)$ ($L \leq_{\mathsf{N}}$ $[\mathrm{Succ}, <]^2$) is not contained in many standard inference types, which include EX , BC and their team variants. Our proof uses decidability results from the theory of ω-automata and we will include them in this section.

In addition, we use an operator approach. We construct an operator, call the *omega operator*, which *translates* appropriate concept classes to witness the corresponding separations. A similar approach was used in [35] in the study of query inference degrees. The decidability results about the query language $[+, <]$ [18] were used in that case.

Informally the omega operator Ω will map each function f into a set A with characteristic string that is of the following form:

$$A = 0^{k_0 \cdot a_0} 1 \cdots 0^{k_n \cdot a_n} 1 \cdots \qquad (k_n \geq 1 \text{ for any } n.) \tag{2}$$

The functional values of f are 'coded' into the sequence $\{k_n\}_{n\geq 0}$. By using the structural properties of a Büchi automaton the sequence $\{a_n\}_{n\geq 0}$ can be computed so that for any set A with characteristic string of the form (2) and for any query $\phi(X) \in [\mathrm{Succ}, <]^2$, $\phi(A)$ can be answered correctly *without* making any mindchanges. We state the following results about Büchi automata which will be used in our proofs.

Definition 7.2 ([15,17]) Let Σ be a finite alphabet and $\sigma, \tau \in \Sigma$. Let P be the infinite string

$$P = \sigma^{p_0}\tau_0 \ldots \sigma^{p_n}\tau_n \ldots ,$$

where $p_0 \geq 0, p_i > 0$ for any $i \geq 1$ and for all i, $\tau_i \in \Sigma - \{\sigma\}$. Define the partial function $thin_{d,\sigma}\colon \Sigma^\omega \to \Sigma^\omega$ as:

$$thin_{d,\sigma}(P) = \sigma^{q_0}\tau_0 \ldots \sigma^{q_n}\tau_n \ldots \ ,$$

where q_i's are defined by the formula

$$q_i = \begin{cases} p_i & \text{if } p_i \le d, \\ \mu q \ [[d < q \le \ d + d!] \ \wedge \ [q \equiv p_i (\text{mod } d!)]] & \text{otherwise.} \end{cases} \tag{3}$$

Theorem 7.3 ([15]) Let Σ be a finite alphabet, $\sigma, \tau \in \Sigma$. Let $\mathcal{A}$ be a Büchi automaton over Σ. Suppose that $\mathcal{A}$ has n states. Let $d \ge 2^{n^2 2^n}$. If

$$X = \sigma^{p_0}\tau_0 \ldots \sigma^{p_n}\tau_n \ldots$$

where $\tau_i \in \Sigma - \{\sigma\}$ for all i. Then X is accepted by $\mathcal{A}$ if and only if $thin_{d,\sigma}(X)$ is accepted by $\mathcal{A}$.

Note 7.4

a) Let $\Sigma = \{0,1\}, \sigma = 0, \tau = 1$. Then $thin_{d,0}(P)$ is defined when P has the property of $x \in P \Rightarrow x+1 \notin P$.

b) Given a query $\phi(X) \in [\text{Succ}, <]^2$, by Theorem 2.7 one can construct an ω automata $\mathcal{A}$ effectively such that $\phi(A)$ is true iff the characteristic strings of A is accepted by $\mathcal{A}$. Theorem 7.3 suggests that if the number of states in $\mathcal{A}$ equals n, then

$$\phi(A) \Leftrightarrow \phi(thin_{d,0}(A)) \qquad \text{for any } d \ge 2^{n^2 2^n}.$$

This fact will be used in our construction of the omega operator.

Now we state the formal definition for the omega operator Ω.

Definition 7.5 Let $\phi_0(X), \phi_1(X), \ldots, \phi_n(X), \ldots$ be a computable enumeration of all queries about a set X in $[\text{Succ}, <]^2$ via a fixed procedure P_1. Let $\mathcal{A}_0, \ldots, \mathcal{A}_n, \ldots$ be the corresponding Büchi automata obtained from the queries via a fixed procedure P_2. (Note: by Theorem 2.7, such a procedure exists). Let n_j ($j \in \mathsf{N}$) be the number of states of $\mathcal{A}_j$. Define

$$h_j = n_j^2 2^{n_j}; \quad d_j = 2^{h_j} \text{ and } \quad m_j = \max \{d_i : 0 \le i \le j\}.$$

We identify a function $f : \mathsf{N} \to \mathsf{N}$ with the string $f(0) \cdots f(n) \cdots \in \mathsf{N}^\omega$ and a set A with its characteristic string. The Omega operator $\Omega : \mathsf{N}^\omega \to \{0,1\}^\omega$ is defined as:

$$\Omega(f) = 0^{l(0,f(0))}1 \cdots 0^{l(n,f(n))}1 \cdots,$$

where $l(n, f(n)) = (f(n) + 1) \cdot m_n!$ for any n.

Notation 7.6 We also use the following notations when using the operator Ω.

a) For any $\mathcal{C} \subseteq \text{DEC}$, $\quad \Omega(\mathcal{C}) = \{\Omega(f) : f \in \mathcal{C}\}$.
b) Ω denotes the transformation which takes a string $f(0) \ldots f(n)$ (identified with $f[n]$) to $\Omega(f[n]) = 0^{l(0,f(0))}1 \ldots 0^{l(n,f(n))}1$.
c) The x^{th} position of the string $\Omega(f[n]) = 0^{l(0,f(0))}1 \ldots 0^{l(n,f(n))}1$ ($0 \le x < \sum_{i=0}^{n} l(i, f(i)) \ + 1$) is denoted by $\Omega(f[n](x))$.

Lemma 7.7 Let $\phi(X)$ be a query in the language $[\text{Succ}, <]^2$ and $f : \mathsf{N} \to \mathsf{N}$. Then there is an effective procedure E such that

a) On input $\phi(X)$, E determines the length of a *finite* initial segment $\alpha(f)$ of f and,
b) the truth of the query $\phi(\Omega(f))$ can be determined effectively from $\alpha(f)$.

Proof: Given a query $\phi \in [\text{Succ}, <]^2$ and a function $f : \mathsf{N} \to \mathsf{N}$, we use the procedure P_1 in Definition 7.5 to enumerate the queries in $[\text{Succ}, <]^2$. The number i such that $\phi = \phi_i$ (syntactically) can be determined. For definiteness we assume that i is the least one. We set $\alpha(f) = \{(0, f(0)), \ldots, (i, f(i))\}$. The following procedure determine the truth of $\phi(\Omega(f))$ from $\alpha(f)$.

Procedure

a) Note that $\Omega(\alpha(f)) = 0^{p_0}1 \ldots 0^{p_i}1$ where $p_k = l(k, f(k)) = (f(k) + 1) \cdot m_k!$ for all $k \le i$. As the m_k's can be effectively determined, hence $p_0, \ldots, p_i$ can be computed from $\alpha(f)$.
b) For $k \le i$, compute q_i using formula (3) from Definition 7.2 and for $k > i$, set $q_k = q_i$.
c) Construct an ω automaton $\mathcal{A}_1$ for the string $(0^{q_1}1 \ldots 0^{q_i}1) \cdot (0^{q_i}1)^\omega$. (Note that this string is definable by a query in $[\text{Succ}, <]^2$).
c) Construct an ω automaton $\mathcal{A}_2$ for the query $\phi(X)$.
d) Construct the intersection of the automata $\mathcal{A}_1$ and $\mathcal{A}_2$.
e) Test the emptiness of $\mathcal{A}_1 \cap \mathcal{A}_2$, that is, to determine if the automata $\mathcal{A}_1 \cap \mathcal{A}_2$ accepts at least one string in $\{0, 1\}^\omega$. This step is effective [2,36].
f) Output TRUE if the intersection is non-empty. Otherwise output FALSE.

End of Procedure

Recall from Definition 7.2 that

$$thin_{d_i,0}(A) = 0^{q_0}1\dots0^{q_n}1\dots\ ,$$

and from part *b*). of Note 7.4,

$$\phi(\Omega(f)) = \phi_i(\Omega(f)) = \phi_i(0^{p_0}1\cdots0^{p_n}1\cdots) = \phi_i(0^{q_0}1\cdots0^{q_n}1\cdots).$$

We shall show that $q_k = q_i$ for any $k \geq i$, which implies

$$(0^{q_1}1\dots0^{q_i}1)\cdot(0^{q_i}1)^\omega = thin_{d_i,0}(\Omega(f)).$$

Let $j > i$. Since $d_i! \mid m_j!$ and for all j's, $k_j \geq 1$ and $d_j > 2$. We have

$$p_j = k_j \cdot m_j! \geq k_j \cdot (d_i!) \geq d_i! > d_i.$$

Therefore, from Definition 7.2 we have

$$q_j = \mu q\ [[d_i < q \leq\ d_i + d_i!]\ \wedge\ [q \equiv p_j (\mathrm{mod}\ d_i!)]].$$

But $q \equiv\ p_j(\mathrm{mod}\ d_i!) \Rightarrow\ q \equiv\ k_j \cdot m_j! \equiv\ 0(\mathrm{mod}\ d_i!)$. Hence for any $j \geq i,\ q_j = d_i!$. This completes the proof. ▮

7.3 Second Case

We will show that for any $L \leq_{\mathsf{N}} [\mathrm{Succ}, <]^2)$,

$$\mathrm{QAN}_0\,([\mathrm{Succ}, <]^2) \not\subseteq [1, n]\mathrm{BC}\,.$$

This comparison result will be further generalized in the next section.

Note 7.8 In Definition 2.4, we define the inference type QAN (L) to be the collection of all classes of *computable functions* where their properties (expressible as queries in L) are learnable. This definition can be easily generalized to classes of *total* functions in the same way. In the following theorem, we implicitly assume that the answer inference type are defined in the general sense.

Theorem 7.9 $\Omega(\mathsf{N}^\omega) \in \mathrm{QAN}_0\,([\mathrm{Succ}, <]^2)$.

$$\begin{tikzcd}
\cdots f(n) \cdots f(0) \arrow[r, "M"] & \cdots e_n \cdots e_0 \arrow[d, "(T_2)"] \\
\cdots \Omega(f)(n) \cdots \Omega(f)(0) \arrow[u, "(T_1)"] \arrow[r, "M'"] & \cdots e_n^{'} \cdots e_0^{'}
\end{tikzcd}$$

Fig. 4. Construction of M' from M

Proof: Follows from Lemma 7.7. ▮

Corollary 7.10 $\Omega(\text{DEC}) \in \text{QAN}_0\left([\text{Succ}, <]^2\right)$.

Proof: Follows from Theorem 7.9. ▮

We now demonstrate that $\text{QAN}_0\left([\text{Succ}, <]^2\right)$ cannot be contained in many standard inference types. In fact, this separation holds for all inference types (for learning programs) that satisfy the following *invariance* property.

Definition 7.11 Let $\mathcal{I}$ be an inference type for inferring programs. $\mathcal{I}$ is *invariant* under Ω if

$$(\forall \mathcal{C} \subseteq \text{DEC})[\Omega(\mathcal{C}) \in \mathcal{I} \Leftrightarrow \mathcal{C} \in \mathcal{I}].$$

Theorem 7.12 The Inference type EX is invariant over Ω.

Proof: Let $\mathcal{C} \subseteq \text{EX}(M)$. We will construct an IIM M' such that $\Omega(\mathcal{C}) \subseteq \text{EX}(M')$. To show this, we first consider the diagram as shown in Figure 4.

Our algorithm for M' first read in a sufficiently long initial segment of $\Omega(f)$ until some initial segment of f are recovered (labeled as T_1 in the diagram). Feed the initial segment of f obtained to machine M. Assuming that the output of M is an index of f, construct an index for $\Omega(f)$ (labeled as T_2). Repeat the process when longer initial segments of $\Omega(f)$ are obtained. It remains to note that given an index of a computable function f, one can obtain an index of the set $\Omega(f)$ effectively. Hence, each conjecture e_i from machine M can be *transformed* to a conjecture $e_i^{'}$ for $\Omega(f)$ and when the conjecture for f from machine M is correct, the corresponding conjecture for $\Omega(f)$ will also be correct. Finally we note that the reverse direction also holds since given an index for $\Omega(f)$, one can also obtain an index for f effectively. ▮

Note 7.13 Clearly, the inference type BC is also invariant under Ω. By applying the same procedure to each machine, we see that the corresponding

Fig. 5. Sample Comparison Results with variants of EX

team's types are also invariant under Ω. The invariance of these inference types are also preserved when we restrict the number of mindchanges. Therefore, the combinations such as $[a,b]\mathrm{EX}_m$ $(a \geq b \geq 1, m \geq 0)$ and etc. are all invariant under Ω. However, it is unclear if the inference types with anomalies are invariant under Ω.

Theorem 7.14 Let $\mathcal{I}$ be invariant under Ω. Then

$$(\forall L \leq_{\mathsf{N}} [\mathrm{Succ}, <]^2)[\mathrm{DEC} \notin \mathcal{I} \Rightarrow \mathrm{QAN}_0\,(L) \not\subseteq \mathcal{I}].$$

Proof: Since $\mathcal{I}$ is invariant under Ω, $\mathrm{DEC} \notin \mathcal{I} \Rightarrow \Omega(\mathrm{DEC}) \notin \mathcal{I}$. However, it follows from Theorem 7.9 that $\Omega(\mathrm{DEC}) \in \mathrm{QAN}_0\,([\mathrm{Succ}, <]^2)$. Hence, $\Omega(\mathrm{DEC}) \in \mathrm{QAN}_0\,([\mathrm{Succ}, <]^2) - \mathcal{I}$. The general case follows from Lemma 2.14. ∎

Corollary 7.15 Let $L \leq_{\mathsf{N}} [\mathrm{Succ}, <]^2$ and $i, j \in \mathsf{N} \cup \{*\}$ $(i \neq 0)$. Suppose $\mathcal{I}$ is invariant under Ω. Then $\mathrm{DEC} \notin \mathcal{I} \Rightarrow \mathrm{Q}_i\mathrm{AN}_j\,(L) \not\subseteq \mathcal{I}$.

Proof: By Theorem 7.14 and the fact that $\mathrm{QAN}_0\,(L) \subseteq \mathrm{Q}_i\mathrm{AN}_j\,(L)$. ∎

By the remarks stated in Note 7.13 and prior results, we can visualize the difference between the learning of answers and the learning of programs (See Figure 5 for sample comparison results with variants of EX). In fact, we have

Corollary 7.16

(1) $L \leq_{\mathsf{N}} [\mathrm{Succ}, <]^2 \Rightarrow (\forall n \geq 1)[\mathrm{QAN}_0\,(L) \not\subseteq [1,n]\mathrm{BC}]$.
(2) For all b, $(\forall n \geq 1)[\mathrm{QAN}_0\,[+, <, \mathrm{POW}_b] \not\subseteq [1,n]\mathrm{BC}]$.

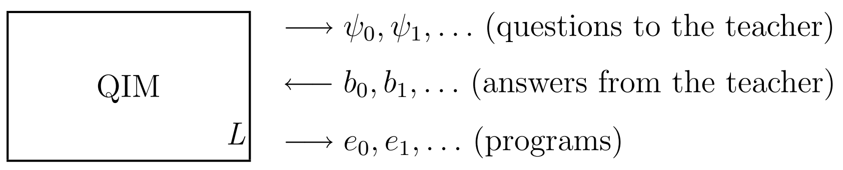
QIM
L
$\longrightarrow \psi_0, \psi_1, \ldots$ (questions to the teacher)
$\longleftarrow b_0, b_1, \ldots$ (answers from the teacher)
$\longrightarrow e_0, e_1, \ldots$ (programs)

Fig. 6. Query Inference Machine

Proof: Since $(\forall n \geq 1)[\mathrm{DEC} \notin [1,n]\mathrm{BC}]$ and $[1,n]\mathrm{BC}$ is invariant under Ω. Part *a* follows immediately. Part *b* follows from Lemma 2.14. ▮

8 Lifting Results to Query Inference Types

In this section, we will explore the relations between query inference machines [19] and answer inference machines. Our discussion leads to a partial answer to one of the central questions in query inference: which relations between inference types remains true for their query counterparts. We will show that for any query languages that are reducible to $[\mathrm{Succ}, <]^2$, separations between many standard inference types remains true for their query versions. This settles some conjectures [17] in query inference.

8.1 Relationships Among Learning Machines

We recall the definition of query inference machines (See Figure 6) and query inference types.

Definition 8.1 ([19])

a) A query inference machine (QIM) is a total Turing machine that tries to infer a program that computes f. It can make queries about the computable function f in a particular query language L (and magically obtain answers). Note that a QIM gets all its information from making queries and does not see the data. However, it can request data such as $f(5)$ by asking questions $f(5) = 0?, f(5) = 1?, \ldots$ until a yes answer is obtained.

b) QEX (L) is the collection of concept classes that can inferred by some QIM M when M makes queries in the query language L and use the learning criteria EX. Generally, if $\mathcal{I}$ is a passive inference type then $\mathrm{Q}\mathcal{I}\,(L)$ will represent the corresponding query inference type. Variants such as teams, anomalies, bounding mindchanges, bounding the number of alternations of quantifiers can be considered likewise.

We observe that the collection of queries a QIM can make is exactly the same collection of queries that some answer inference machine wish to learn. The following lemma state a useful relationship between query inference and inductive inference. The intuition is that when the collection of queries that a QIM can make is decidable, then the QIM is nothing more than an IIM. Queries made to the teacher (See Figure 6) can be *decided* correctly by an AIM. More generally, if the answers to the collection of queries can be learned, then many concepts that can be learned by some QIM's can also be learned by some other IIM's.

Lemma 8.2 Let L be any reasonable query language and $\mathcal{C} \subseteq \mathrm{DEC}$.

a) Suppose that $\mathcal{I}$ is a passive inference type for learning programs. Then

$$\mathcal{C} \in \mathrm{Q}\mathcal{I}\,(L) \wedge\ \mathcal{C} \in \mathrm{QAN}_0\,(L) \Rightarrow \mathcal{C} \in \mathcal{I}\,.$$

b) Suppose that $\mathcal{J}$ is an inference type that tries to learn programs in the limit and $\mathrm{Q}\mathcal{J}\,(L,*)$ be the corresponding query inference types that ask finitely many queries, then

$$\mathcal{C} \in \mathrm{Q}\mathcal{J}\,(L,*) \wedge\ \mathcal{C} \in \mathrm{QAN}\,(L) \Rightarrow \mathcal{C} \in \mathcal{J}\,.$$

Proof:

a) Suppose that $\mathcal{C} \subseteq \mathrm{Q}\mathcal{I}\,(L)$ via M and $\mathcal{C} \subseteq \mathrm{QAN}_0\,(L)$ via M'. Then all the answers to those queries made by M while inferring programs for functions in $\mathcal{C}$ can be obtained by using the AIM M'. It is clear that there is an IIM M'' which combines the use of M and M'.

b) The answer of those queries made by M while inferring programs for functions in $\mathcal{C}$ can be learned by using some AIM M'. Hence the action of the QIM can be simulated by restarting computations every time when a different answers to some queries are obtained. As the answers of the queries will stabilize eventually to the correct set of answers and the QIM make only finitely many queries, the simulation will eventually succeed in learning a correct program.

▮

Daley [11] shows that it is always possible to trade machines for errors in BC style learning, namely,

Lemma 8.3

$$(\forall n \geq 1)(\forall a \geq 0)[\,[1,n]\mathrm{BC}^a \subseteq [1,n(a+1)]\mathrm{BC}\,].$$

Hence Corollary 7.16 can be further improved even though the inference type $[1,n]\mathrm{QBC}^a\,(L)$ may not be invariant under Ω.

Theorem 8.4 Let $a, n \in \mathsf{N}$ and $n \geq 1$. Then

$$\mathrm{QAN}_0\,([\mathrm{Succ},<]^2) \not\subseteq\; [1,n]\mathrm{QBC}^a\,([\mathrm{Succ},<]^2)$$

Proof: Assume by way of contradiction that

$$\mathrm{QAN}_0\,([\mathrm{Succ},<]^2) \subseteq\; [1,n]\mathrm{QBC}^a\,([\mathrm{Succ},<]^2),$$

where $a, n \in \mathsf{N}$ $(n \neq 0)$. Hence we have $(\mathrm{QAN}_0\,([\mathrm{Succ},<]^2) \subseteq \mathrm{QAN}_0\,([\mathrm{Succ},<]^2)) \wedge (\mathrm{QAN}_0\,([\mathrm{Succ},<]^2) \subseteq [1,n]\mathrm{QBC}^a\,([\mathrm{Succ},<]^2))$.

By Lemma 8.2, we obtain

$$\mathrm{QAN}_0\,([\mathrm{Succ},<]^2) \subseteq [1,n]\mathrm{BC}^a\,.$$

In addition, by trading machines for errors (Lemma 8.3) we get

$$\mathrm{QAN}_0\,([\mathrm{Succ},<]^2) \subseteq [1,n(a+1)]\mathrm{BC}\,.$$

By Corollary 7.10 $\Omega(\mathrm{DEC}) \in \mathrm{QAN}_0\,([\mathrm{Succ},<]^2)$, hence $\Omega(\mathrm{DEC}) \in [1,n(a+1)]\mathrm{BC}$. Since $[1,n(a+1)]\mathrm{BC}$ is invariant under Ω, $\mathrm{DEC} \in [1,n(a+1)]\mathrm{BC}$, which is a contradiction (See [9]). ▮

Corollary 8.5 Given a query language L,

$$L \leq_{\mathsf{N}} [\mathrm{Succ},<]^2 \Rightarrow (\forall n \geq 1)(\forall a \in \mathsf{N})[\mathrm{QAN}_0\,(L) \not\subseteq\; [1,n]\mathrm{QBC}^a\,(L)].$$

Proof: Use reduction in Corollary 8.4. ▮

Note 8.6 We use only the fact that $(\forall a \in \mathsf{N})[\mathrm{DEC} \notin [1,a]BC]$. Hence we obtain a short proof of the result ([17])

$$L \leq_{\mathsf{N}} [\mathrm{Succ},<]^2 \Rightarrow (\forall n \geq 1)(\forall a \in \mathsf{N})[\mathrm{DEC} \notin [1,n]\mathrm{QBC}^a\,(L)].$$

8.2 A Lifting Lemma

Lemma 8.7 Let $\mathcal{I}$ be a passive inference type that is invariant under Ω. Then

$$A \in \mathrm{Q}\mathcal{I}\,([\mathrm{Succ}, <]^2) \text{ iff } A \in \mathcal{I}.$$

Proof: Assume $A \in \mathrm{Q}\mathcal{I}\,([\mathrm{Succ}, <]^2)$. By Lemma 7.7 we can take the query-inference procedure for A and turn it into a passive-inference procedure for $\Omega(A)$. Hence $\Omega(A) \in \mathcal{I}$. Since $\mathcal{I}$ is invariant we have $A \in \mathcal{I}$.

It is clear that if $A \in \mathcal{I}$ then $A \in \mathrm{Q}\mathcal{I}\,([\mathrm{Succ}, <]^2)$. ▮

Theorem 8.8 Let $\mathcal{I}$, $\mathcal{J}$ be two passive inference types and $\mathrm{Q}\mathcal{I}\,(L)$, $\mathrm{Q}\mathcal{J}\,(L)$ be the corresponding query inference types. Suppose $\mathcal{I}$ and $\mathcal{J}$ are *invariant* under Ω. Then

$$\mathcal{J} - \mathcal{I} \neq \emptyset \quad \Rightarrow \quad \mathcal{J} - \mathrm{Q}\mathcal{I}\,([\mathrm{Succ}, <]^2) \neq \emptyset.$$

Proof: Suppose $\mathcal{J} - \mathcal{I} \neq \emptyset$. Hence there exists $\mathcal{C} \subseteq \mathrm{DEC}$ such that $\mathcal{C} \in \mathcal{J} - \mathcal{I}$. By Lemma 8.7, $\mathcal{C} \notin \mathrm{Q}\mathcal{I}\,([\mathrm{Succ}, <]^2)$. Hence $\mathcal{J} - \mathrm{Q}\mathcal{I}\,([\mathrm{Succ}, <]^2) \neq \emptyset$. ▮

The following theorem is proven by a sight modification of Theorem 8.8

Theorem 8.9 Let $\mathcal{I}$, $\mathcal{J}$ be two passive inference types and $\mathrm{Q}\mathcal{I}\,(L)$, $\mathrm{Q}\mathcal{J}\,(L)$ be the corresponding query inference types. Suppose $\mathcal{I}$ and $\mathcal{J}$ are *invariant* under Ω. Let $L \leq_{\mathsf{N}} [\mathrm{Succ}, <]^2$. Then

$$\mathcal{J} - \mathcal{I} \neq \emptyset \quad \Rightarrow \quad \mathcal{J} - \mathrm{Q}\mathcal{I}\,(L) \neq \emptyset.$$

Note 8.10 For many passive inference types $\mathcal{I}$ and $\mathcal{J}$ and languages L, the inclusion $\mathrm{Q}\mathcal{I}\,(L) \subseteq \mathrm{Q}\mathcal{J}\,(L)$ follows immediately from the proof of the inclusion $\mathcal{I} \subseteq \mathcal{J}$. Hence, for such cases we actually have

$$\mathcal{I} \subset \mathcal{J} \quad \Rightarrow \quad \mathrm{Q}\mathcal{I}\,(L) \subset \mathrm{Q}\mathcal{J}\,(L).$$

Theorem 8.9 helps to lift separation results to their query versions. For instance, Smith [30] shows that

Theorem 8.11 ([30])

$$\mathrm{EX} \subset [1,2]\mathrm{EX} \subset [1,3]\mathrm{EX} \subset \cdots \quad \text{and} \quad \mathrm{BC} \subset [1,2]\mathrm{BC} \subset [1,3]\mathrm{BC} \subset \cdots$$

All these inference types are invariant under Ω. Therefore we have the following corollary, which settles a few questions raised in [17,20].

Corollary 8.12 Let $L \leq_{\mathsf{N}} [\mathrm{Succ}, <]^2$. Then

a) $\mathrm{QEX}\,(L) \subset [1,2]\mathrm{QEX}\,(L) \subset [1,3]\mathrm{QEX}\,(L) \subset \cdots$
b) $\mathrm{QBC}\,(L) \subset [1,2]\mathrm{QBC}\,(L) \subset [1,3]\mathrm{QBC}\,(L) \subset \cdots$

Proof:

$$\mathrm{QEX}\,(L) \subseteq [1,2]\mathrm{QEX}\,(L) \subseteq [1,3]\mathrm{QEX}\,(L) \subseteq \cdots$$

and

$$\mathrm{QBC}\,(L) \subseteq [1,2]\mathrm{QBC}\,(L) \subseteq [1,3]\mathrm{QBC}\,(L) \subseteq \cdots$$

follows directly from their definitions. It suffices to note that by applying lifting lemma (Theorem 8.9) to Theorem 8.11, the non-inclusion follows immediately. ∎

Interestingly, we also have an analogue of the '$[24,49]$' Theorem [12] for query inference types.

Corollary 8.13 Let $L \leq_{\mathsf{N}} [\mathrm{Succ}, <]^2$. Let c, d be such that $24/49 < c/d < 1/2$. Then

a) $[1,2]\mathrm{QEX}_0\,(L) \subset [2,4]\mathrm{QEX}_0\,(L) \subset [24,49]\mathrm{QEX}_0\,(L)$.
b) $[24,49]\mathrm{QEX}_0\,(L) - [c,d]\mathrm{QEX}_0\,(L) \neq \emptyset$.

Proof:

a) It is easy to see that

$$[1,2]\mathrm{QEX}_0\,(L) \subseteq [2,4]\mathrm{QEX}_0\,(L) \subseteq [24,49]\mathrm{QEX}_0\,(L).$$

By applying the lifting lemma (Theorem 8.9) to the results ([12])

$$[2,4]\mathrm{EX}_0 - [1,2]\mathrm{EX}_0 \neq \emptyset$$

and

$$[24,49]\mathrm{EX}_0 - [2,4]\mathrm{EX}_0 \neq \emptyset,$$

we have

$$[1,2]\mathrm{QEX}_0\,(L) \subset [2,4]\mathrm{QEX}_0\,(L) \subset [24,49]\mathrm{QEX}_0\,(L).$$

b) By applying the lifting lemma (Theorem 8.9) to the following result [12]

$$[24,49]\mathrm{EX}_0 - [c,d]\mathrm{EX}_0 \neq \emptyset$$

we have

$$[24,49]\mathrm{QEX}_0\,(L) - [c,d]\mathrm{QEX}_0\,(L) \neq \emptyset$$

▮

Corollary 8.14

a) For all b,

$\mathrm{QEX}\,([+,<,\mathrm{POW}_b]) \subset [1,2]\mathrm{QEX}\,([+,<,\mathrm{POW}_b]) \subset [1,3]\mathrm{QEX}\,([+,<,\mathrm{POW}_b]) \subset [1,4]\mathrm{QEX}\,([+,<,\mathrm{POW}_b]) \subset \cdots$

b) For all b,

$\mathrm{QBC}\,([+,<,\mathrm{POW}_b]) \subset [1,2]\mathrm{QBC}\,([+,<,\mathrm{POW}_b]) \subset [1,3]\mathrm{QBC}\,([+,<,\mathrm{POW}_b]) \subset [1,4]\mathrm{QBC}\,([+,<,\mathrm{POW}_b]) \subset \cdots$

c) Let $b \in \mathbb{N}$. Let $c,d \in \mathbb{N}$ be such that $24/49 < c/d < 1/2$. Then

$$\begin{aligned}[1,2]\mathrm{QEX}_0\,([+,<,\mathrm{POW}_b]) &\subset [2,4]\mathrm{QEX}_0\,([+,<,\mathrm{POW}_b]) \\ &\subset [24,49]\mathrm{QEX}_0\,([+,<,\mathrm{POW}_b])\end{aligned}$$

and $\quad [24,49]\mathrm{QEX}_0\,([+,<,\mathrm{POW}_b]) - [c,d]\mathrm{QEX}_0\,([+,<,\mathrm{POW}_b]) \neq \emptyset$.

Proof: This follows from Corollaries 8.12, 8.13, and Fact 2.12. ▮

9 Open Problems

Query languages: The results in this work holds for languages that are reducible to $[\mathrm{Succ},<]^2$. Our techniques depend on decidability results from the theory of omega automata. Do these results still hold for any query languages with a *decidable* base languages?

Lifting separations: We proved a lifting lemma which 'lifts' every separations to their query inference analogue when no anomalies are involved. Will the lifting lemma hold when anomalies are allowed ?

Lifting subset relations: Can subset relations be also lifted? For example, can one show the following:

$$(\forall L \leq_{\mathsf{N}} [\mathrm{Succ},<]^2)(\forall a,b,c,d \in \mathsf{N})$$

$$[a,b]\mathrm{EX}_0 \subseteq [c,d]\mathrm{EX}_0 \Rightarrow [a,b]\mathrm{QEX}_0\,(L) \subseteq [c,d]\mathrm{QEX}_0\,(L)$$

Acknowledgment: We thank We thank Kalvis Apsitis, John Case, Mark Changizi, Lazaros Kikas, April Lee, Georgia Martin, and Frank Stephan for proofreading. We thank John Case, Jim Royer, and Frank Stephan for discussion of the primitive recursive recursion theorem.

References

[1] J. Büchi. Weak second order arithmetic and finite automata. *Zeitsch. f. math. Logik und Grundlagen d. Math.*, 6:66–92, 1960.

[2] J. Büchi. On a decision method in restricted second order arithmetic. In E. Nagel et al., editors, *Logic Methodology and Philosophy of Science*, pages 1 –11. Stanford University Press, 1962.

[3] D. Angluin and C. Smith. Inductive inference: Theory and methods. *Computing Surveys*, 15:237–269, 1983.

[4] G. Baliga, J. Case, S. Jain, and M. Suraj. Machine learning on higher-order programs. *The Journal of Symbolic Logic*, 59(2):486–500, 1994.

[5] Ben-David. Can finite samples detect singularities. *Algorithmica*, 22, 1998. Prior version in STOC92.

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

[7] J. Case, S. Jain, and A. Sharma. On learning limiting programs. *Internat. J. Found. Comput. Sci*, 3(1):93–115, 1992.

[8] J. Case, E. Kinber, A. Sharma, and F. Stephan. On the classification of computable languages. *Information and Computation*, 192, 2004.

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

[10] Y. Choueka. Theory of automata on ω-tapes. *Journal of Computer and System Sciences*, 8:117–142, 1974.

[11] R. Daley. On the error correcting power of pluralism in bc-type inductive inference. *Theoretical Computer Science*, 24(1):95–104, 1983.

[12] R. Daley, B. Kalyanasundaram, and M. Velauthapillai. Breaking the probability 1/2 barrier in fin-type learning. *Journal of Computer and System Sciences*, 50:574–599, 1995.

[13] M. Davis. Hilbert's 10^{th} problem is unsolvable. *Amer. Math. Monthly*, 80:233–269, 1973.

[14] M. Davis, P. H., and R. J. The decision problem of exponential diophantine equations. *Ann. math.*, 74:425–436, 1961.

[15] C. Elgot and M. O. Rabin. Decidability and undecidability of extensions of second (first) order theory of (generalized) successor. *The Journal of Symbolic Logic*, 31:169–181, 1966.

[16] H. B. Enderton. *A mathematical introduction to logic.* Academic Press, 1972.

[17] W. Gasarch and G. Hird. Automata techniques for query inference machines. *Annals of pure and applied logic*, 117:171–203, 2002. Prior version in *C*omputational Learning Theory, 1995 (COLT).

[18] W. Gasarch, M. Pleszkoch, and R. Solovay. Learning via queries in $[+, <]$. *Journal of Symbolic Logic*, 57:58–81, 1992.

[19] W. Gasarch and C. Smith. Learning via queries. *Journal of ACM*, 39:649–676, 1992.

[20] W. Gasarch and C. H. Smith. A survey of inductive inference with an emphasis on queries. In A. Sorbi, editor, *Complexity, Logic, and Recursion Theory*, number 187 in Lecture notes in Pure and Applied Mathematics Series. M. Dekker., 1997.

[21] W. I. Gasarch, M. G. Pleszkoch, F. Stephan, and M. Velauthapillai. Classification using information. *Annals of mathematics and artificial intelligence*, 23(1-2), 1998. Earlier version in ALT94.

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

[23] S. Jain, D. Osherson, J. Royer, and A. Sharma. *Systems that Learn: An Introduction to Learning Theory for Cognitive and Computer Scientists.* M.I.T. Press, 1999.

[24] K. T. Kelly. *The logic of reliable inquiry.* Logic and Computation in Philosophy. The Clarendon Press Oxford University Press, New York, 1996.

[25] Y. V. Matiyasevich. Enumerable sets are diophantine. *Doklady Academy Nauk. SSSR*, 191:279–282, 1970. Translation in Sov. Math. Dokl. 11 (1970), 354-357.

[26] Y. V. Matiyasevich. *Hilbert's tenth problem.* Foundations of Computing Series. MIT Press, Cambridge, MA, 1993. Translated from the 1993 Russian original by the author, With a foreword by Martin Davis.

[27] M. Rabin. Decidable theories. In J. Barwise, editor, *Handbook of Mathematical Logic.* North Holland, 1977.

[28] J. Royer and J. Case. *Subrecursive Programming Systems: Complexity and Succinctness.* Birkhauser, 1994.

[29] D. Siefkes and J. R. Büchi. The monadic second order theory of all countable ordinals. In G. Muller and D. Siefkes, editors, *Decidable Theories II*, volume 328 of *Lecture Notes in Mathematics.* Springer Verlag, New York, 1973.

[30] C. Smith. The power of pluralism for automatic program synthesis. *Journal of ACM*, 29(4):1144–1165, 1982.

[31] C. Smith, R. Wiehagen, and T. Zeugmann. Classification of predicates and languages. In *Computational learning theory : EuroCOLT '93*, pages 171–181. Oxford University Press, 1994.

[32] C. H. Smith. *A Recursive Introduction of Theory of Computation.* Springer Verlag, 1994.

[33] C. H. Smith and R. Wiehagen. Generalization versus classification. *Journal of Experimental and Theoretical Artificial Intelligence*, 7:163–174, 1995.

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

[35] F. Stephan. Learning via queries and oracles. *Annals of Pure and Applied Logic*, 94:273–296, 1998. Prior version in *C*omputational Learning Theory, 1995 (COLT).

[36] W. Thomas. Automata on infinite objects. In J. V. Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume 2, chapter 4, pages 135 –191. Elsevier Science Publishers B.V., 1990.

[37] W. Thomas. Language, automata and logic. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages Vol.3 BEYOND WORDS*, pages 389–455. Spring Verlag, 1997.