

# Learning via Finitely Many Queries

Andrew C. Lee

University of Louisiana at Lafayette<sup>1</sup>

## Abstract

This work introduces a new query inference model that can access data and communicate with a teacher by asking finitely many boolean queries in a language  $L$ . In this model the parameters of interest are the number of queries used and the expressive power of  $L$ . We study how the learning power varies with these parameters. Preliminary results suggest that this model can help studying query inference in an resource bounded environment.

## 1 Introduction

In computability-theoretic learning theory, concepts are modelled as collections of computable functions from  $\mathbb{N}$  to  $\mathbb{N}$ , where  $\mathbb{N}$  is the set of all natural numbers. An inference machine  $M$  is simply a *total* Turing machine. Given a concept  $\mathcal{S}$  and  $f \in \mathcal{S}$ ,  $M$  takes finite initial segments  $\{(0, f(0)), \dots, (n, f(n))\}$  of  $f$  (abbrev. as  $\sigma_n(f)$ ) as input and outputs (the Gödel number of) a program that tries to compute  $f$ .  $f$  is said to be *EX*-learnable by  $M$  [Gol67] if

$$\lim_{n \rightarrow \infty} M(\sigma_n(f)) = e \tag{1}$$

and that the program  $e$  computes  $f$ .  $\mathcal{S}$  is said to be *EX*-learnable by  $M$  when every  $f \in \mathcal{S}$  is *EX*-learnable by  $M$ . The object

$$EX = \{\mathcal{S} : (\exists M)[\mathcal{S} \text{ is } EX\text{-learnable by } M]\} \tag{2}$$

is the *inference type* associated to the learning criteria *EX*. Denote  $M(\sigma_n(f))$  by  $e_n$ . Suppose that the following condition (i.e. (3)) which is less restrictive than condition (1), is satisfied:

$$(\exists n_0)(\forall n)[(n \geq n_0) \rightarrow \text{program } e_n \text{ computes } f] \tag{3}$$

Then  $M$  does *learn* the function  $f$  *semantically* although the limit  $\lim_{n \rightarrow \infty} e_n$  may not exist. The learning model that uses criteria (3) is referred as *behavioral correct learning*. Its inference type is denoted by *BC*. Note that both *EX* and *BC* (and many other basic inference criteria [CS83] as well) learners learn *passively* by observing data.

By contrast, an *active* way of learning is to ask questions. Gasarch and Smith [GS92] studied learning via queries in the computability-theoretic setting. Roughly speaking, a *query inference machine* (abbrev. as QIM) is an inference machine that learn concepts by actively asking questions to a teacher, who will provide correct answers to these questions. Questions are boolean queries formulated in a query language  $L$  about the function  $f$  currently under investigation. These questions represent the extra knowledge we can have while learning functions from the given concept. For any *passive* inference type (i.e. inference without using queries)  $\mathcal{I}$  (e.g.  $\mathcal{I} = EX, BC$  etc.), the corresponding query inference type is  $Q\mathcal{I}(L)$ :

$$Q\mathcal{I}(L) = \{\mathcal{C} : (\exists M)[\mathcal{C} \text{ is } \mathcal{I}\text{-learnable by an QIM } M \text{ via the query language } L]\} \tag{4}$$

The query language  $L$  is a parameter that may affect the learning power. In this work we *refine* the query inference model. We introduce the *bounded query* inference types  $Q\mathcal{I}(L)\langle k \rangle$ . The quantity

---

<sup>1</sup>Computer Science Department, University of Louisiana at Lafayette, Box 41771, Lafayette, Louisiana 70504-1771  
Email: cxl9999@louisiana.edu

$k$  ( $k \in \mathbb{N}$ ) denotes the number of queries allowed, is another parameter of interest. The additional learning power gained by a query inference machine can be measured either *quantitatively* by the number of queries made or *qualitatively* by the expressive power of the language  $L$ .

The main point of this paper is to introduce this framework and demonstrate its potential for measuring the two types of resources mentioned above in query inference. Our preliminary results try to answer the following basic questions:

- *Does more queries always help?*: We show that (Corollary 3.6 and 3.7) for many passive inference types  $\mathcal{I}$ , more queries always help and this fact is *independent* of the query languages used.
- *Can one trade ‘quantity’ for ‘quality’?*: We show that in our query inference model (Corollary 3.8) enriching a query language cannot reduce the number of queries used in the learning process.

Asking  $k$  boolean questions has at most  $2^k$  possible sequence of answers. Hence any inference process that uses  $k$  boolean queries can be simulated via a team of  $2^k$  inference machines (See Section 3 for definitions related to team inference). It is natural to ask the following question:

- *Can team inference be simulated by bounded query inference?*

In particular, we ask if a team inference process by  $2^k$  learners (without using queries) can be simulated by a single machine using at most  $k$  boolean queries in a language  $L$ . We show that when using the language  $[+, \times]$  one can simulate the team learning process. However, it asks questions that are unrelated to the concepts it wish to learn. For query languages that are reducible to  $[Succ, <]^2$  ([GH95, GL97]), our results indicate that a bounded query inference machine cannot even simulate a team of size two, regardless of the number of questions asked.

There has been much research on computability-theoretic learning theory. We refer the interested readers to [AS83, JORS99, Smi94, GS95, GS97] and the references cited therein for surveys of many major developments in this research area.

## 2 Notation and Basic Definitions

We assume familiarity with the basic notions of logic ([End72]) and computability theory ([Rog67, Soa87]).  $\{\varphi_e\}_{e \in \mathbb{N}}$  denotes an arbitrary acceptable numbering of all but only partial computable functions. Throughout this paper **COMP** denotes the collection of all computable functions from  $\mathbb{N}$  to  $\mathbb{N}$ . **COMP**<sub>0,1</sub> denotes the set when we restrict to  $\{0, 1\}$  valued computable functions. We identify it with the collection of all computable subsets of  $\mathbb{N}$ .

### 2.1 Query Languages and Inference Models

A *query language*  $L$  consists of the usual logical symbols with equality, quantifiers, symbols for first order variables, symbols for every element of  $\mathbb{N}$  and a special symbol  $\mathcal{F}$  denoting the function we wish to learn. Extra symbols for some functions and relations on  $\mathbb{N}$  may be included, and the query language will be denoted by these symbols. For example,  $[<]$  denotes the query language with the relation  $<$  and  $[+, <]$  denotes the query language with the relation  $<$  and the function  $+$ . We use  $[\emptyset]$  to denote the query language with no extra symbols. Most of the query languages we consider are first order and we will use a superscript 2 when we allow set variables and their quantifications. For example,  $[Succ, <]^2$  is the query language with the successor function ( $Succ(x) = x + 1$ ) and the relation  $<$ . In addition, it consists of set variables and allows quantifications over these set variables. We use small (resp. large) letters are used for number (resp. set) variables, which range over  $\mathbb{N}$  (resp. subsets of  $\mathbb{N}$ ). We will consider the language  $[Succ, <]^2$  in Section 4.

Throughout this paper,  $L$  denotes a *reasonable* query language. That is, all the symbols in  $L$  represent computable operations. A few examples of queries in various languages are given below:

| $L$           | query   | interpretation                              |
|---------------|---|---|
| $[\emptyset]$ | $(\forall y)(\exists x)[f(x) = y]$  | ‘Is $f$ surjective?’                        |
| $[<]$         | $(\exists x)(\forall y)[x < y \rightarrow f(x) = 0]$                                    | ‘Does $f$ have finite support?’             |
| $[+, <]$      | $(\exists x)(\exists p)(\forall y)[(x < y) \wedge (p > 0) \rightarrow f(y) = f(y + p)]$ | ‘Is $f$ eventually periodic?’               |
| $[+, \times]$ | ‘Is $x \in K$ ?’  | ‘Does $\varphi_x$ converges on input $x$ ?’ |

Note that  $K$  is the *halting set*<sup>2</sup>. Queries about sets can be similarly defined. Given a query language  $L$ , one may *enrich* the language to  $L'$  by introducing more extra symbols to  $L$ . We will call  $L'$  an extension of  $L$ . For any two query languages  $L_1, L_2$ , when all the extra symbols in  $L_1$  can be interpreted in the language  $L_2$ , then  $L_2$  is at least as expressive as  $L_1$ . For any query languages  $L_1$  and  $L_2$ , denote  $L_1 \preceq L_2$  if  $L_2$  is at least as expressive as  $L_1$ . It is known [End72] that  $[\emptyset] \preceq [<] \preceq [+, <] \preceq [+, \times]$ . Also,  $[Succ, <]$ <sup>2</sup> is at least as expressive as  $[Succ, <]$  because we allow set variables and their quantifications.

In the original model [GS92], a query inference machine obtains all of its information from making queries to a teacher and cannot access the data directly. It can however request whatever data its wants via queries (e.g., a QIM can find out what  $f(13)$  is by asking the questions ‘ $f(13) = 0$ ?’, ‘ $f(13) = 1$ ?’, etc. until a YES answer is obtained). In this work we consider an QIM which can access data and post a *finite* number of queries.

**Definition 2.1** Let  $k \in \mathbb{N} \cup \{*\}$ . Let  $\mathcal{M}_k$  be the collection of QIMs which can access the data values but is allowed to post  $k$  logical queries in  $L$ . The *bounded query inference type*  $Q\mathcal{I}(L)\langle k \rangle$  is defined as the set

$$Q\mathcal{I}(L)\langle k \rangle = \{C : (\exists M \in \mathcal{M}_k)[C \text{ is } \mathcal{I}\text{-learnable by } M]\}$$

Note that  $Q\mathcal{C}(L)\langle 0 \rangle = C$  and  $Q\mathcal{I}(L)\langle * \rangle$  denote the inference type when the machine can ask finitely many questions. Also,  $Q\mathcal{C}(L)\langle * \rangle \subseteq Q\mathcal{C}(L)$ . It follows that we have the following inference type hierarchy:

$$Q\mathcal{I}(L)\langle 0 \rangle \subseteq Q\mathcal{I}(L)\langle 1 \rangle \subseteq \dots \subseteq Q\mathcal{I}(L)\langle k \rangle \subseteq \dots \subseteq Q\mathcal{I}(L)\langle * \rangle \quad (5)$$

### 3 Main Results

Let  $\mathcal{I}$  be any inference criteria and let  $1 \leq k \leq n$ . The *team inference type*  $[k, n]\mathcal{I}$  ([Smi82]) is defined as the set

$$\{C : (\exists M_1) \dots (\exists M_n)(\forall f \in C)[\text{at least } k \text{ out of } n \text{ } M_i\text{'s learns } f \text{ via the criteria } \mathcal{I}]\} \quad (6)$$

Note that for any  $f_1, f_2 \in C$  ( $f_1 \neq f_2$ ), the sub-collection of machines that learn each of them may be different.

Observe that asking  $k$  questions has at most  $2^k$  possible sequence of answers. Hence any inference process that uses  $k$  boolean queries can be simulated via a team of  $2^k$  inference machines. That is,

<sup>2</sup>The query language  $[+, \times]$  can express the question ‘ $x \in A$ ?’ for any computably enumerated set  $A$  [Mat93]. Note the use of this result in query inference [GS92].

**Theorem 3.1** For any passive inference type  $\mathcal{I}$  and for any  $k \geq 1$ ,  $Q\mathcal{I}(L)\langle k \rangle \subseteq [1, 2^k]\mathcal{I}$ .

Our main result compares the bounded query inference types with team inference types that *allow errors*. Let  $a \in \mathbb{N}$  and  $f, g : \mathbb{N} \rightarrow \mathbb{N}$ . We write  $f =^a g$  when the set  $\{x : f(x) \neq g(x)\}$  has at most  $a$  elements. An inference machine  $M$  is said to  $BC^a$ -learns  $f$  ([CS83, Smi82]) if  $M$  fed  $f$  outputs over time an infinite sequence of programs  $p_0, p_1, \dots$ , such that for all but finitely many  $n$ ,  $\varphi_{p_n} =^a f$ . The inference type  $BC^a$  is the set

$$BC^a = \{\mathcal{S} : (\exists M)[\mathcal{S} \text{ is } BC^a\text{-learnable by } M]\} \quad (7)$$

By combining (6) and (7) one can consider the team inference types  $[k, n]BC^a$  ( $1 \leq k \leq n$ ). We also need the following technical definitions in our proof.

**Definition 3.2** Let  $f : \mathbb{N} \rightarrow \mathbb{N}$  and  $\tau$  be a finite initial segment of  $f$ . Let  $\text{dom}(\tau)$  (resp.  $\text{rng}(\tau)$ ) denote the domain (resp. range) of  $\tau$ .  $\phi_k(f)$  ( $k \geq 1$ ) denotes the query

$$\phi_k(f) = (\forall y)(\exists x_1)(\exists x_2)\dots(\exists x_k)[\bigwedge_{i \neq j} (x_i \neq x_j) \bigwedge_{i \geq 1}^k f(x_i) = y]. \quad (8)$$

We say that  $f$  is *k-fold surjective* if  $\phi_k(f)$  is true. In addition,  $f$  is *exactly k-fold surjective* if  $\phi_k(f)$  is true but  $\phi_{k+1}(f)$  is false. We also use the notation  $\phi_k(\tau)$ . We say that  $\phi_k(\tau)$  is true if

$$(\forall y \in \text{rng}(\tau))(\exists x_1 \in \text{dom}(\tau))\dots(\exists x_k \in \text{dom}(\tau))[\bigwedge_{i \neq j} (x_i \neq x_j) \bigwedge_{i \geq 1}^k f(x_i) = y]. \quad (9)$$

**Fact 3.3** The query  $\phi_k(f)$  (resp.  $\phi_k(\sigma)$ ) is true implies  $\phi_j(f)$  (resp.  $\phi_j(\sigma)$ ) are true for all  $j \leq k$ .  $\phi_k(f)$  ( $\forall k \in \mathbb{N}$ ) does not use any extra symbols and hence it can be expressed in *any* query languages.

The following are generalizations of notions introduced in [Smi82].

**Definition 3.4** Let  $f : \mathbb{N} \rightarrow \mathbb{N}$ ,  $x \in \mathbb{N}$  and let  $Y$  be an *infinite* computable subset of  $\mathbb{N}$ . We use  $n_Y(x)$  to denote the smallest element in  $Y$  that is greater than  $x$ .

- a) The *dips* of  $f$  in  $Y$  is the set  $D_Y(f) = \{n_Y(x) : x \in Y \wedge f(n_Y(x)) < f(x)\}$ .
- b) If  $\sigma$  and  $\tau$  are initial segments of  $f$  where  $\tau$  extends  $\sigma$ , then  $\tau$  *monotonically extends*  $\sigma$  along  $Y$  if  $f$  is monotonic nondecreasing when restricted to the domain  $[\text{dom}(\tau) - \text{dom}(\sigma)] \cap Y$ .

**Theorem 3.5** For any query language  $L$  and for any  $k \geq 1$ ,

$$QEX(L)\langle k \rangle - \bigcup_{a=0}^{\infty} [1, 2^k - 1]BC^a \neq \emptyset.$$

**Proof:** It suffices to show that for any  $k \geq 1$  and  $a \geq 0$ , there is a concept  $\mathcal{S}(k, a)$  such that  $\mathcal{S}(k, a) \in QEX([\emptyset])\langle k \rangle$  but  $\mathcal{S}(k, a) \notin [1, 2^k - 1]BC^a$ .

*Construction of  $\mathcal{S}(k, a)$ :* Partition  $\mathbb{N}$  into  $2^n$  infinite computable subsets  $Y_1, Y_2, \dots, Y_{2^n}$  of  $\mathbb{N}$ . By an implicit use of the recursion theorem (See [CR94]), we may let  $\mathcal{C}_i = \{f \in \mathbf{COMP} : f \text{ is exactly } i\text{-fold surjective and } \varphi_f(\max(D_{Y_i}(f)) = f)\} (\neq \emptyset)$  and we set  $\mathcal{S}(k, a) = \mathcal{C}_1 \cup \dots \cup \mathcal{C}_{2^n}$ . By Definition 3.2,  $\mathcal{C}_i$  and  $\mathcal{C}_j$  ( $i \neq j$ ) are pairwise disjoint. Moreover, for any  $f \in \mathcal{S}(k, a)$  and  $i$  ( $1 \leq i \leq 2^k$ ), we have the property that

$$\phi_i(f) \text{ is true iff } f \in \mathcal{C}_i \cup \mathcal{C}_{i+1} \cup \dots \cup \mathcal{C}_{2^k} \quad (10)$$

$\mathcal{S}(k, a) \in QEX([\emptyset]\langle k \rangle)$ : By property (10) a bounded query inference machine  $M$  can first apply binary search via the queries  $\phi_1, \dots, \phi_{2^k}$  to determine the unique  $i$  such that  $f \in \mathcal{C}_i$ . This step takes at most  $k$  queries selected from  $\phi_1, \dots, \phi_{2^k}$ , and all of them can be formulated in the query language  $[\emptyset]$ . Machine  $M$  will then proceed by reading initial segments of  $f$ . It will first output  $f(y_0)$  where  $y_0$  is the first element of  $Y_i$  (Note:  $Y_i$  computable) and  $M$  will update its output to output  $f(n_{Y_i}(x))$  when  $M$  reads an  $x \in Y_i$  such that  $f(n_{Y_i}(x)) < f(x)$ . It follows that  $M$ 's outputs will converge to the value  $f(\max(D_{Y_i}(f)))$ , which is (by definition of  $\mathcal{C}_i$ ) an index of a function that computes  $f$ .

$\mathcal{S}(k, a) \notin [1, 2^k - 1]BC^a$ : We present the proof for the case when  $k = 2$ , the general case follows similar ideas. Let  $M_1, M_2$  and  $M_3$  be a team of inference machines. We will construct a program  $\varphi_{e_4}$  by diagonalization against  $M_1, M_2, M_3$  such that

1. If  $\varphi_{e_4}$  is total, then the function (say  $g$ ) computed by  $\varphi_{e_4}$  is an element in  $\mathcal{C}_4$  but  $g$  cannot be  $BC^a$ -learned by  $M_1, M_2$  nor  $M_3$ .
2. If  $\varphi_{e_4}$  is not total, then it follows from the construction that one machine, say  $M_1$ , cannot learn any function that extends an initial segment (say  $\sigma$ ) of  $\varphi_{e_4}$ . By using  $\sigma$  and a similar construction which diagonalizes against the two remaining machines (say  $M_2$  and  $M_3$ ), one can create another program  $\varphi_{e_3}$ . Note that the number of machines involved is reduced by 1. If  $\varphi_{e_3}$  is total, then the function (say  $h$ ) computed by  $\varphi_{e_3}$  is an element in  $\mathcal{C}_3$  but  $h$  cannot be  $BC^a$ -learned by  $M_2$  nor  $M_3$  (and definitely not by  $M_1$ ). If  $\varphi_{e_3}$  is not total, one can reiterate the steps and construct another program  $\varphi_{e_2}$ .
3. When the number of machines involved drops to zero, we can simply take  $\varphi_{e_1}$  to be a surjective extension that monotonically extends the current segment in all  $Y_i$ 's.

To simplify our presentation, we will present the construction of program  $\varphi_{e_4}, \varphi_{e_3}$  and the transition steps only.

**Program  $\varphi_{e_4}$ :**

**Initialization** Let  $y$  be the smallest element in  $Y_4$ . Set  $\varphi_{e_4}^0 = \sigma_0$  where  $(y, e_4) \in \sigma_0$  and  $\phi_4(\sigma_0)$  is true. Put the machines  $M_1, M_2, M_3$  in a priority queue. and let  $M_1$  be the front of the queue. We will use  $\varphi_{e_4}^s$  to denote the initial segment constructed at the beginning of stage  $s$ .

**Stage  $s$  :**

1. Let  $M$  be the machine that is at the front of the queue. Search for extensions  $\varphi_{e_4}^s \subseteq \sigma_s \subset \tau_s$  and  $x_1, \dots, x_{a+1} \in \text{dom}(\tau_s - \sigma_s)$  that satisfy the following collection of requirements:
  - a) (Have at least  $a + 1$  errors)  $\forall i \leq a + 1, M(\sigma)(x_i) \neq \tau_s(x_i)$ .
  - b) (Preserves  $f(\max(D_{Y_4}(f))) = e_4$ )  $\tau_s$  monotonic extends  $\varphi_{e_4}^s$  along  $Y_4$ .
  - c) (Avoids making  $f$  5-fold surjective)  $\phi_5(\tau_s)$  is false.
2. We reach this step if the previous search terminates. To preserve 4-fold surjectivity, we extend  $\tau_s \subseteq \tau$  such that  $\phi_4(\tau)$  is true and  $\phi_5(\tau)$  is false.
3. (Reset parameters and queues) Set  $\varphi_{e_4}^{s+1} = \tau$ . Put  $M$  to the end of the priority queue and go to stage  $s + 1$ .

**End of Program  $\varphi_{e_4}$**

If  $\varphi_{e_4}$  is total and let  $f = \varphi_{e_4}$ . By construction  $f \in \mathcal{C}_4$  and  $f \notin BC^a(M_1, M_2, M_3)$ . If  $\varphi_{e_4}$  is not total, then we perform the following *transition steps*. Let  $s$  be the least stage that the search does not terminate. Let  $\sigma = \varphi_{e_4}^s$  and  $M_1$  be the machine at the front of the queue at stage  $s$ . Note that for any extension  $\tau$  of  $\sigma$  that satisfy conditions 1b). and 1c). in the code of  $\varphi_{e_4}$ ,  $M_1(\tau)$  can converge on at most  $a$  points not in  $\text{dom}(\sigma)$ . hence  $M_1$  cannot  $BC^a$ -identify any function in  $\mathcal{C}$  that monotonically extends  $f$  along  $Y_4$ . By another implicit use of the recursion theorem, we obtain the following program  $e_3$ .

**Program**  $\varphi_{e_3}$ :

**Initialization** Let  $z$  be the smallest element that are in the set  $Y_3 - \text{dom}(\varphi_{e_4}^s)$ . Set  $\varphi_{e_3}^0 = \sigma_0^3$ , where  $\varphi_{e_4}^s \subset \sigma_0^3$ ,  $\phi_3(\sigma_0^3)$  is true and  $(z, e') \in \sigma_0^3$ . Here  $e'$  is a *padded version* of  $e_3$  (i.e.:  $\varphi_{e_3} = \varphi_{e'}$  where  $e'$  is large enough to satisfy the conditions stated above). Remove  $M_1$  from the queue. Put the remaining machines in a priority queue. and let  $M_2$  be the front of the queue. (i.e. Now the queue is  $M_2, M_3$ ).

**Stage**  $s$  :

1. Let  $M$  be the machine that is at the front of the queue. Search for extensions  $\varphi_{e_3}^s \subseteq \sigma_s \subset \tau_s$  and  $x_1, \dots, x_{a+1} \in \text{dom}(\tau_s - \sigma_s)$  that satisfy the following collection of requirements:
  - a) (Have at least  $a + 1$  errors)  $\forall i \leq a + 1, M(\sigma)(x_i) \neq \tau_s(x_i)$ .
  - b) (Preserves  $f(\max(D_{Y_3}(f))) = e_3$ )  $\tau_s$  monotonic extends  $\varphi_{e_3}^s$  along  $Y_3$  and  $Y_4$ .
  - c) (Avoids the function constructed to become 4-fold surjective)  $\phi_4(\tau_s)$  is false.
2. Note that We reach this step if the previous search terminates. (Preserve 3-fold surjectivity) We extend  $\tau_s \subseteq \tau$  such that  $\phi_3(\tau)$  is true and  $\phi_4(\tau)$  is false.
3. (Reset parameters and queues) Set  $\varphi_{e_3}^{s+1} = \tau$ . Put  $M$  to the end of the priority queue and go to stage  $s + 1$ .

**End of Program**  $\varphi_{e_3}$

■

**Corollary 3.6** let  $\mathcal{I} = EX$  or  $BC$ . Then for any query languages  $L$  and for any  $k \in \mathbb{N}$ ,

- a).  $Q\mathcal{I}(L)\langle k \rangle \subset Q\mathcal{I}(L)\langle k + 1 \rangle$  and b).  $Q\mathcal{I}(L)\langle k \rangle \subset Q\mathcal{I}(L)\langle * \rangle$ .

**Proof:** Part b). follows from part a). For part a), For any  $k, a \geq 0$ ,

$$Q\mathcal{I}(L)\langle k \rangle \subseteq [1, 2^k]\mathcal{I} \quad (\text{Thm 3.1}) \quad \& \quad [1, 2^k]\mathcal{I} \subset [1, 2^{k+1} - 1]BC^a \quad ([\text{Smi82}]) \quad (11)$$

By Theorem 3.5 and the fact that  $EX \subseteq BC$ , there is a concept  $\mathcal{C} \in Q\mathcal{I}(L)\langle k + 1 \rangle - [1, 2^{k+1} - 1]BC^a$  and by (11)  $\mathcal{C}$  cannot be in  $Q\mathcal{I}(L)\langle k \rangle$ . ■

**Corollary 3.7** Let  $L$  be a query language and  $k \in \mathbb{N}$ . Let  $r \geq 1$  and  $\mathcal{I} = EX^r$  or  $\mathcal{I} = BC^r$ . We have a).  $Q\mathcal{I}(L)\langle k \rangle \subset Q\mathcal{I}(L)\langle k + 1 \rangle$  and b).  $Q\mathcal{I}(L)\langle k \rangle \subset Q\mathcal{I}(L)\langle * \rangle$ .

**Proof:** It suffices to note that when  $a \geq r$ , equation (11) also holds for  $\mathcal{I} = EX^r$  or  $\mathcal{I} = BC^r$  and the previous arguments also follow in these cases. ■

We also shows that improving the quality of boolean queries cannot reduce the number of queries made. An intuitive explanation is that any  $k$  boolean queries can *describe* no more than  $2^k$  bits of information.

**Corollary 3.8** Let  $L$  be any query language and let  $\mathcal{I} = EX$  or  $BC$ . For any extension  $L'$  of  $L$  and any  $k \in \mathbb{N}$ ,  $Q\mathcal{I}(L)\langle k + 1 \rangle \not\subseteq Q\mathcal{I}(L')\langle k \rangle$ .

**Proof:** We prove the case when  $\mathcal{I} = EX$ . By Theorem 3.5,  $(\exists \mathcal{C} \in \mathbf{COMP})[(\mathcal{C} \in QEX(L)\langle k + 1 \rangle - [1, 2^{k+1} - 1]BC^a)]$ . For any extension  $L'$  of  $L$ , If  $\mathcal{C} \in QEX(L')\langle k \rangle \Rightarrow \mathcal{C} \in [1, 2^k]EX \subset [1, 2^{k+1} - 1]EX$ . A contradiction. ■

In the original query inference model, the inference process may require potentially infinitely many queries. We will show that when the language  $L$  can express the ' $<$ ' relation, asking only finitely many questions restrict the learning power of the query inference machine. We generalize the techniques in Theorem 3.5 to separate the query inference types  $QEX([\langle \cdot \rangle])\langle * \rangle$  and  $QEX([\langle \cdot \rangle])$ .

**Theorem 3.9**  $QEX([\lt]) - QEX([\lt])\langle * \rangle \neq \emptyset$

**Proof:** We sketch the main ideas only. First by Theorem 3.1  $QEX([\lt])\langle * \rangle \subseteq \bigcup_{n=1}^{\infty} [1, n]EX$ . We partition  $\mathbf{N}$  into  $Y_i$ 's ( $i \geq 1$ ), where all of them are infinite and computable. Let  $\mathcal{C} = \bigcup_{n=1}^{\infty} \mathcal{C}_n$ , where  $\mathcal{C}_n$  is the class of computable functions  $f$  such that  $\varphi_{f(\max_{D_{Y_n}(f)})} = f$  and  $f$  is *eventually*  $n$ -fold surjective, that is, if the following query

$$\phi_k(f) = (\exists z)(\forall y)(\exists x_1)(\exists x_2) \dots (\exists x_k)[z < y \rightarrow \bigwedge_{i \neq j} (x_i \neq x_j) \bigwedge_{i \geq 1}^k f(x_i) = y].$$

is true. One can show that  $\mathcal{C} \in QEX([\lt]) - \bigcup_{n=0}^{\infty} [1, n]EX$  by a minor modification of the technique as in the proof of Theorem 3.5. ■

## 4 Relationships with Team Inference

Theorem 3.1 states that for any passive inference type  $\mathcal{I}$  and for any  $k \geq 1$ ,  $Q\mathcal{I}(L)\langle k \rangle \subseteq [1, 2^k]\mathcal{I}$ . If the equality  $Q\mathcal{I}(L)\langle k \rangle = [1, 2^k]\mathcal{I}$  holds for some  $L$ , the language  $L$  is expressive enough so that the team learning process can be simulated by an active learning strategy via  $k$  queries in  $L$ .

We first show that when  $L = [+ , \times]$ , the above equality holds.

**Theorem 4.1**  $(\forall n \in \mathbf{N})[QEX([+ , \times])\langle n \rangle = [1, 2^n]EX]$ .

**Proof:** We only sketch the proof here. We claim that  $(\forall n \in \mathbf{N})[QEX([+ , \times])\langle n \rangle \supseteq [1, 2^n]EX]$ . Since all computably enumerable set are diophantine [Mat93, GS92], In  $[+ , \times]$  one can ask the queries ‘Does there exists a machine  $i < j$  so that  $M_i$  learns the function  $f$ ?’. By combining this observation with binary search, one can first determine which machine can actually learns  $f$  and then simulate that machine. ■

However, for many other languages used in query inference, the answer is negative. We will treat  $L$  as a parameter and apply the technique of reduction to deal with this parameter (See [GH95, GL97]). We only state which languages are reducible to  $[Succ, <]^2$  here. We referred the interested readers to [GH95] for a detailed discussions on various types of reductions of query languages used in our inference model.

**Theorem 4.2** [GH95] Let  $b \geq 2$ . Let

1.  $POW_b$  be the unary predicate that determines if a number in the set  $\{b^n | n \in \mathbf{N}\}$
2.  $POLY_b$  be the unary predicate that determines if a number in the set  $\{n^b | n \in \mathbf{N}\}$
3.  $FAC$  be the unary predicate that determines if a number in the set  $\{n! | n \in \mathbf{N}\}$

The Languages  $[+ , <]$ ,  $[+ , < , POW_b]$ ,  $[Succ , < , POW_b]^2$ ,  $[Succ , < , POLY_b]^2$  and  $[Succ , < , FAC]^2$  are reducible to  $[Succ , <]^2$ .

**Theorem 4.3**  $(\forall k \geq 1)[[1, 2]EX - QEX([Succ , <]^2)\langle k \rangle] \neq \emptyset$ .

**Proof:** (sketch) Let  $\mathcal{C} \in [1, 2]EX - EX$ . By the lifting theorem [GL97], there exists an operator  $\Omega$  such that  $\Omega(\mathcal{C}) \in [1, 2]EX$  and suppose  $\Omega(\mathcal{C}) \in QEX([Succ , <]^2)\langle k \rangle$ , then  $\Omega(\mathcal{C}) \in QEX([Succ , <]^2)$  which in turn implies [GL97]  $\Omega(\mathcal{C}) \in EX$ , a contradiction. ■

**Corollary 4.4** For any  $L$  that is reducible to  $[Succ, <]^2$  and for any  $k \geq 1$ ,

$$[1, 2]EX - QEX(L)\langle k \rangle \neq \emptyset.$$

**Proof:** By applying reductions (See [GH95]) as all the languages stated above were shown to be reducible to  $[Succ, <]^2$ . ■

## 5 Open problems

A new query inference model that can access data and communicate with a teacher by asking a *bounded* number of logical queries are studied. The following questions are of interest:

- Simulation of a team learning process via active learning strategies:

Can one develop a languages  $L$  so that  $Q\mathcal{I}(L)\langle k \rangle = [1, 2^k]\mathcal{I}'$ ? We showed that when  $L = [+ , \times]$ , equality do holds. However, in our proof the learner asks questions that are unrelated to the functions that we wish to learn. It will be of interest to find a  $L$  such that the equality holds and queries in  $L$  *truly* asks about *properties* of the functions the machine are observing.

- Using *non-boolean* queries:

No matter how expressive a boolean query can be, it only allow the communication of a single bit of information. Developing new bounded query inference models that use non-boolean queries are also of interest.

**Acknowledgement:** The author wish to thank William Gasarch, Steele Russell for comments on earlier drafts of this work.

## References

- [AS83] D. Angluin and C.H. Smith. Inductive inference: Theory and methods. *Computing Surveys*, 15:237–269, 1983.
- [CR94] John Case and James Royer. *Subrecursive Programming Systems: Complexity and Succinctness*. Springer Verlag, 1994.
- [CS83] J. Case and C.H. Smith. Comparison of identification criteria for machine inductive inference. *Theoretical Computer Science*, 25:193–220, 1983.
- [End72] H. B. Enderton. *A mathematical introduction to logic*. Academic Press, 1972.
- [GH95] W.I. Gasarch and G.H. Hird. Reduction for learning via queries. In *Proceedings of 8<sup>th</sup> Annual ACM Conference on Computational Learning Theory*, pages 152–161, 1995.
- [GL97] W.I. Gasarch and Andrew C.Y. Lee. Inferring answers to queries. In *Proc. 10<sup>th</sup> Annual ACM Conference on Computational learning theory*, pages 275–284. COLT, 1997.
- [Gol67] E.M. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.
- [GS92] W.I. Gasarch and C.H. Smith. Learning via queries. *Journal of ACM*, 39:649–676, 1992.
- [GS95] W. I. Gasarch and C. H. Smith. Recursion theoretic models of learning: some results and intuitions. *Annals of Mathematics and Artificial Intelligence*, 15(2):155–166, 1995.
- [GS97] W.I. 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.

- [JORS99] Sanjay Jain, Daniel Osherson, James Royer, and Arun Sharma. *Systems that Learn: An Introduction to Learning Theory*. MIT Press, Cambridge, MA., second edition edition, 1999.
- [Mat93] Yuri 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.
- [Rog67] H Rogers, Jr. *Theory of recursive functions and effective computability*. McGraw-Hill, 1967.
- [Smi82] C.H. Smith. The power of pluralism for automatic program synthesis. *Journal of ACM*, 29(4):1144–1165, 1982.
- [Smi94] C. H. Smith. Three decades of team learning. In *Proc. 5th Int. Workshop on Algorithmic Learning Theory*, pages 211–228. Springer-Verlag, 1994.
- [Soa87] R.I. Soare. *Recursively Enumerable Sets and Degrees*. Omega Series. Springer-Verlag, 1987.