A Survey of Inductive Inference with an Emphasis on Queries

William Gasarch*
University of Maryland

Carl H. Smith†
University of Maryland

1 Introduction

If I said to you $2, 4, 6, 8$, then you might reply to me in any of the following ways.

1. The next value is 10.
2. The formula is $f(x) = 2x$.
3. Who do we appreciate?[1]

Let us assume you made the second response. More generally, your goal is to find out what function I am giving you. What if I now continue the

*Dept. of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. (gasarch@cs.umd.edu).

Supported in part by NSF grant CCR-9301339.

†Dept. of Computer Science, University of Maryland, College Park, MD 20742. On leave at the University of Amsterdam (smith@cs.umd.edu).

Supported in part by NSF grant CCR-9301339.

[1]In America, if children wanted to praise (say) Frank Stephan for showing that there is a low set A such that $REC \in BC[A]$, they might say, "2,4,6,8, who do we appreciate? Frank, Frank, yea, Frank!"

sequence with $8, 8, 8$ (so the entire sequence thus far is $2, 4, 6, 8, 8, 8, 8$). Then you might change your mind and guess that the function I am giving you is

$$f(x) = \begin{cases} 2x & \text{if } x = 1, 2, 3, 4; \\ 8 & \text{otherwise.} \end{cases}$$

It turns out that you are correct. The only data you will see from now on are $8, 8, \ldots$. However, note that you are not sure that you are correct.

If I had told you ahead of time that the function I am giving you will eventually be constant, then you can be somewhat confident (but not totally) of your final guess. If I told you that the first number you see repeated is the function value from that point on, then you are totally confident in your final guess. If I had told you nothing about the function ahead of time, then I can foil any guess you make.

Informally, I have a class of recursive functions $\mathcal{S}$ that I will tell you something about (I cannot lie). I will then show you $f(0), f(1), \ldots$ where f is some function in $\mathcal{S}$. You will try to figure out the code for f. If you eventually figure it out, then you are happy. Here are some examples.

1. $\mathcal{S}$ is the class of functions that are cubic polynomials with integer coefficients. After seeing $f(0), f(1), f(2), f(3)$ you can easily determine which polynomial it is by interpolation.

2. $\mathcal{S}$ is the class of functions f such that $f(3)$ *is* a program for f. You think I'm nuts, and there cannot be such a function. I remind you that by the Recursion Theorem there are infinitely many functions in $\mathcal{S}$. When you see $f(0), f(1), f(2)$ you do not make a guess as to the function. But when you see $f(3)$ you guess $f(3)$ and you are right.

3. $\mathcal{S}$ is the class of primitive recursive functions. It turns out that you *can* tell what the function is from seeing data. But the algorithm takes full advantage of the phrase 'eventually figure it out.' See Example 3.3.

In this survey we will formally define many variants of inferring a class of functions. We emphasize those variants where the learner can ask questions as well as see data. Our context is recursion theoretic.

In this survey we emphasis the types of mathematical techniques used in proofs. We include proof sketches but very few actual proofs. The term

'proof sketch' is somewhat ambiguous; however, suffice it to say that some of our sketches are are sketchier than others.

There have been other surveys of some of this material. A survey of inductive inference that included this material and models that were not recursion theoretic is [AS83]. A survey concentrating on team inference is [Smi94b]. A survey that looks at the features of learning that are most similar to human learning is [GS95].

2 Standard Notation

We use standard notation from recursion theory. For an elementary introduction to recursion theory see [Smi94a]. For a more advanced treatment see [Soa87].

Notation 2.1 The natural numbers are defined by $\{0, 1, 2, \ldots\}$ and are denoted by N. We denote subsets of N by capital letters (usually A or B) and elements of N by small letters (usually a, b, c, d, n, m).

Notation 2.2 Throughout this paper $M_0, M_1, \ldots$ is a standard list of all Turing machines, $M_0^{()}, M_1^{()}, \ldots$ is a standard list of all oracle Turing machines, $\varphi_0, \varphi_1, \ldots$ is the acceptable programming system, obtained by letting φ_e be the partial recursive function computed by M_e. We refer to e as a *program* or *index*.

Notation 2.3 The expression $M_e(x)\downarrow$ means that the computation $M_e(x)$ converges. The expression $M_e(x)\uparrow$ means that the computation $M_e(x)$ diverges.

Notation 2.4 Let $M_{e,s}$ be the machine that, on input x, runs $M_e(x)$ for s steps, outputs $M_e(x)$ if the computation has halted within s steps, and diverges otherwise. Let $\varphi_{e,s}$ be the partial function computed by $M_{e,s}$.

Notation 2.5 A' is the halting problem relative to A, that is, $\{e : M_e^A(e)\downarrow\}$. A is *high* if $\emptyset'' \leq_{\mathrm{T}} A'$. A is *low* if $A' \leq_{\mathrm{T}} K$.

Definition 2.6 REC is the class of all recursive functions. $RECSET$ is the class of all recursive sets.

Notation 2.7 Let σ, τ be strings over an alphabet Σ. $|\sigma|$ denotes the length of σ. $\sigma \preceq \tau$ means that σ is a prefix of τ. We think of σ as being a map from $\{0, 1, \ldots, |\sigma| - 1\}$ to Σ. If $a \in \mathsf{N}$, then we use σa^ω to denote the total function whose characteristic string has initial segment σ and then consists of all a's. The most common usage of this notation will be to refer to the function $\sigma 0^\omega$.

Notation 2.8 Let $\sigma \in \{0,1\}^*$ and $M^{()}$ be an oracle Turing machine. M^σ is the Turing machine that attempts to simulate $M^{()}$ by answering questions as though σ were an initial segment of the oracle. If ever a query is made that is bigger than $|\sigma| - 1$, then the computation diverges. Any divergent computation that results from running $M^\sigma(x)$ is denoted by $M^\sigma(x)\uparrow$.

Notation 2.9 $(\exists^\infty x)$ means 'for an infinite number of x.' $(\forall^\infty x)$ means 'for all but a finite number of x;' equivalently, 'for almost all x.'

The remaining recursion-theoretic notation is from [Soa87].

Notation 2.10 We refer to a set of natural numbers as a *set*. We refer to a set of sets of natural numbers as a *class*. We refer to a set of classes as a *collection*.

3 Definitions in Inductive Inference

We briefly review concepts from inductive inference. For a fuller treatment see [CS83, OSW86b].

Definition 3.1 An inductive inference machine (IIM) is a total Turing machine M. We interpret M as trying to learn a recursive function f as follows. M is presented with the values $f(0)$, $f(1)$, $f(2), \ldots$ and will output conjectures e_n (on input $f(0), \ldots, f(n)$) indicating that M thinks f is computed by φ_{e_n} (based on the data that M has seen so far). Note that φ_{e_n} need not

be a total function. If φ_{e_n} is partial, then M's guess is wrong. The guess may be $\perp$, which indicates that M has no conjecture at this time. (After at least one 'real' guess is made, however, the machine is not permitted to output $\perp$.). M has no other way of obtaining additional information about f. (See Figure 1.)

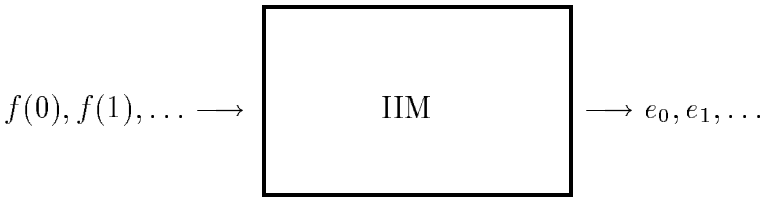

Figure 1

Definition 3.2 Let M be an IIM and f be a function. For all n, let $M(f(0), \ldots, f(n)) = e_n$. If there is an e and an m such that, for all $n \geq m$, $e_n = e$, then M *converges to* e *on input* f. If, additionally, $\varphi_e = f$ then M *inferred* the function f. Let $\mathcal{S} \subseteq REC$. M *infers* $\mathcal{S}$ if it infers every $f \in \mathcal{S}$. EX is the collection of $\mathcal{S} \subseteq REC$ which are inferred by some IIM. The term EX stands for 'explains.' The idea is that we are looking at data and we wish to explain it by producing an index for a machine that generates the data. (This motivation is from [CS83, Put75].)

Example 3.3

1. Let $PRIMREC$ be the class of primitive recursive functions. The class $PRIMREC$ is in EX via the procedure described as follows. Let $q_1, q_2, q_3, \ldots$ be a uniformly recursive enumeration of $PRIMREC$. Upon seeing $f(0)$, ..., $f(n)$ output a program for the least i such that $q_i(0) = f(0), \ldots, q_i(n) = f(n)$. Eventually the correct index will be output forever. Note that the learner does not know when the correct index is being output.

2. $\{f : \varphi_{f(0)} = f\} \in EX$. Upon seeing $f(0)$ just output $f(0)$ and never change your mind. Note that this class of functions is non-empty and quite large by the Recursion Theorem.

3. $\{f : (\forall^\infty x)[f(x) = 0]\} \in EX$. Upon seeing $\sigma = \tau 0^i$ output code for the function that is $\tau 0^\omega$. Eventually the correct index will be output forever. Note that the learner does not know when the correct index is being output.

Note 3.4 The method of inference in example 1 is called *enumeration*. The method of inference in example 2 has no formal name but could be called *coding the program into the function*. In nearly all work in inductive inference these are the only two types of inference used. Barzdins [Bar80] conjectured that these might be the only types of inference there are (formalizing this conjecture is non-trivial). The conjecture was disproved [Ful90b]. An exact characterization of EX (and BC which will be described in Definition 3.6) is in [Wie78b, Wie78a, FKW95].

Definition 3.5 Let $a \in \mathsf{N}$. If f and g are functions, then $f =^a g$ means that f and g differ on at most a places. We may also say that f is *an a-variant of g*. The expression $f =^* g$ means that f and g differ on a finite number of places. We may also say that f is *a finite variant of g*.

We define several variations on inference that have received considerable attention in the literature.

Definition 3.6 Let M be an IIM, f be a recursive function, $c, d, n \in \mathsf{N}$ with $c, d \geq 1$, and $a \in \mathsf{N} \cup \{*\}$.

1. *Mindchanges* [CS83]. If M infers f and only at most n times outputs a conjecture that is different from the most recent previous conjecture, then *M infers the function f with $\leq n$ mindchanges*. $\mathcal{S} \in EX_n$ if there is an IIM M such that, for every $f \in \mathcal{S}$, M infers f with at most n mindchanges. A change from $\perp$ to a real guess is not counted as a mindchange. (The class EX_0 has also been called FIN in the literature.)

2. *Anomalies* [CS83]. Let $a \in \mathsf{N}$. If M is trying to infer f and, from some point on, always outputs e where φ_e is an a-variant of f, then M *a-infers f*. If M is trying to infer f and, from some point on, always outputs e where φ_e is a finite variant of f, then M *$*$-infers f*. In either case ($a \in \mathsf{N}$ or $a = *$) $\mathcal{S} \in EX^a$ if there exists an IIM M such that, for every $f \in \mathcal{S}$, M a-infers f.

3. *Teams* [OSW86a, Smi82, Smi94b]. Let $M_1, \ldots, M_d$ be a set of d IIMs. If at least c of $M_1, \ldots, M_d$ correctly EX-infer f, then f *is* $[c,d]$*-inferred by* $\{M_1, \ldots, M_d\}$. $\mathcal{S} \in [c,d]EX$ if there exist $M_1, \ldots, M_d$ such that, for every $f \in \mathcal{S}$, f is $[c,d]$*-inferred by* $\{M_1, \ldots, M_d\}$. The set of machines $\{M_1, \ldots, M_d\}$ is referred to as a *team*. Note that different functions in $\mathcal{S}$ may be inferred by different size-c subsets of $\{M_1, \ldots, M_d\}$.

4. *Probabilistic Inference* [Fre79, WFK84, Pit89]. Let p be a real number such that $0 \leq p \leq 1$. Let M be a total Turing machine that can also flip coins (this can easily be made rigorous [Gil77]). f is *inferred with probability* p *by* M if the probability that M infers f is at least p. $\mathcal{S} \in EX\langle p \rangle$ if there exists a coin flipping total Turing machine M such that, for every $f \in \mathcal{S}$, M infers f with probability p. (To define this formally requires talking about sample spaces of infinite sequences of coin flips. We will not need such detail.)

5. *Popperian Inference* [CS83]. $\mathcal{S} \in PEX$ if $\mathcal{S} \in EX$ via an IIM that only conjectures total programs. (By contrast, the guesses made by a machine for EX inference may be non-total.) This concept was studied and named in [CS83].

6. *Next Value* [Pod74, CS83]. $\mathcal{S} \in NV$ if there is a machine that, given $f(0), \ldots, f(n-1)$ tries to guess $f(n)$, and is right all but a finite number of times. This corresponds to the case where if I say '2,4,6,8', then you say '10.' However, it turns out to be equivalent to PEX (implicit in [Pod77, Pod75] but see [CS83] for first formal proof). We will not be mentioning it again.

7. *Behaviorally Correct* [Bar74, CS83]. M *behaviorally infers* f if, when M is fed f, eventually all of M's guesses about f are indices e where $\varphi_e = f$. This differs from EX in that if M EX-infers f, then past some point all the guesses are *the same* and compute f, whereas here they need not be the same, they need only compute f. BC is the collection of $\mathcal{S} \subseteq REC$ which can be behaviorally inferred by some IIM M. The key difference between EX and BC is that EX is syntactic (we care about what the program *is*) and BC is semantic (we care about what the program *does*).

8. *Combinations.* The parameters above can be combined to yield inference types like $[c,d]EX^a_n$. Comparisons between these inference types have been studied extensively [FSV89]. We will deal with combinations like PEX_n, PEX^a_n, EX^*_0, and $[1,d]BC^a$. For $[1,d]BC^a$ we will not allow $a = *$, since the class of all recursive functions is in BC^* [CS83]. (We prove this in Theorem 4.1.)

Definition 3.7 We will refer to EX, BC, etc. as *inference types.*

Example 3.8

1. Let $a \in \mathsf{N} \cup \{*\}$. $\{f : \varphi_{f(0)} =^a f\} \in EX^a_0$. Upon seeing $f(0)$ just output $f(0)$ and never change your mind.

2. $\{0^\omega\} \cup \{0^i 1^\omega : i \in \mathsf{N}\} \in EX_1$. First guess that the function is all 0's. If a 1 is spotted in position $i+1$ then guess that the function is $0^i 1^\omega$.

3. $\{0^\omega\} \cup \{0^i 1^\omega : i \in \mathsf{N}\} \cup \{0^i 1^j 0^\omega : i,j \in \mathsf{N}\} \in EX_2$. We leave this to the reader.

4. $\{f : \varphi_{f(0)} =^1 f\} \in [1,2]EX$. We define M_1 and M_2 such that $f \in [1,2]EX$ via M_1 and M_2. M_1 thinks that $\varphi_{f(0)} = f$ and hence guesses $f(0)$ forever. M_2 is sure that $\varphi_{f(0)} =^{=1} f$ (exactly one mistake). Upon seeing $f(0), \ldots, f(s)$ M_2 will compute $\varphi_{f(0),s}(x)$ for every $x \leq s$. If there exists $i \leq s$ such that $\varphi_{f(0),s}(i) \downarrow \neq f(i)$ then let i_0 be that i. If no such i exists, then let $i_0 = \mu x[x \leq s \wedge \varphi_{f(0),s}(x) \uparrow]$ (Such an i_0 exists under the convention that $\varphi_{e,s}(x)$ diverges on all $x \geq s$, which we adopt.) In either case output a program for g such that

$$g(x) = \begin{cases} \varphi_{f(0)}(x) & \text{if } x \neq i_0 \\ f(i_0) & \text{if } x = i_0. \end{cases}$$

In the above definitions we were only concerned with learning *recursive functions.* There is another field concerned with learning *languages* (i.e., r.e. sets). The scenario is that the elements of the set are given in some order, and the learner tries to determine a grammar (i.e., r.e. index) for the set. This study is very different since if (say) 17 has not entered the set

yet, you do not know if it ever will; by contrast, when learning (say) a 0-1 valued recursive function f, you eventually know $f(17)$. Language learning was first defined in [Gol67], but was later developed more in [BB75, CL82, OW82]. Language learning with a team is discussed in [JS90, JS93c, JS95a]. Restrictions on strategies for language learning are discussed in [Ful90a, JS96, KS95, OSW82]. Restrictions on the type of languages to be learned are surveyed in [ZL95]. A book that covers much of this material is [OSW86b].

There are several other bells and whistles one can use to define additional aspects of learning. We list several here, broken up into several categories.

Change in the Machine Model

1. What if your inductive inference machine could ask queries about the function being inferred [GH95, GKP$^+$95, GPS92, GS92, Ste95]?
2. What if your inductive inference machine could ask queries to an auxiliary oracle [FJG$^+$94, GP89, JS93b, KS96, Ste95]?
3. What if your inductive inference machine has limited memory [FKS95]?
4. What if the number of mindchanges is allowed to be an ordinal [FS93]?
5. What if you want to infer functions with domain N but range the rationals [CM95] or the reals [AFS95]?

Variations on the output

1. What if the number of errors allowed is an ordinal [FS93]?
2. What if you allow the number of errors to be infinite [GSSV92, Roy86, SV90]?
3. What if you want to learn a minimal (or near minimal) index for a program [Che82, FJ95, Jai95, JS94b]?
4. If you do not need to learn the function, but merely to classify it in some way, is this any easier [GPSVar, WS95]?

Variations on the Input

1. How much does it help learning to be given some additional information [BC94, JS93a]?

2. What if the information presented to the learner is not quite accurate [FJ89, Jai94]?

3. Can you learn something by the process of elimination [FKS94]?

Misc.

1. If $\mathcal{I}$ and $\mathcal{J}$ are two inference types, and $\mathcal{I} \not\subseteq \mathcal{J}$, then is there a way to measure the strength of that non-inclusion [Kum94]?

2. Given a class of functions $\mathcal{S}$, let's look at the class of all inductive inference machines that learn it [FS97].

3. Is there a way to say that $\mathcal{S}_1$ is harder to infer than $\mathcal{S}_2$ [FKS96, JS94a, JS95b]?

4. How do you choose inductive inference machines to form a team [AFS]?

5. Is there a way to measure the complexity of inference (similar to Blum Complexity for computation) [DS86]?

4 Results in Inductive Inference

We first look at how much power is needed before REC can be inferred.

Theorem 4.1 *Let $a \in \mathsf{N}$ and $m \geq 1$.*

1. *$REC \notin BC$ [Bar74, CS83].*

2. *$REC \notin BC^a$ [CS83].*

3. *$REC \notin [1,m]BC$ [Smi82].*

4. *$REC \notin [1,m]BC^a$ [Smi82].*

5. *$REC \in BC^*$. (This last result is credited to Leo Harrington in [CS83].)*

Proof sketch:

$REC \notin BC$: Let M be an IIM. We construct a partial recursive function f such that either (1) f is actually total recursive, and f is not inferred by M, or (2) f is a finite segment and $f0^\omega$ (which is clearly recursive) is not inferred by M.

We construct f in stages. Assume that at the end of stage s we have the finite initial segment σ_s (i.e., $\sigma_s \prec f$). Look for τ and x such that $\sigma_s \preceq \tau$, $x > |\tau|$, and $\varphi_{M(\tau)}(x) \downarrow$. If such is found, then extend σ_s to some τ' such that $\tau \prec \tau'$, $|\tau'| = x + 1$, and $\tau'(x) \neq \varphi_{M(\tau)}(x)$.

If every stage terminates, then f is total recursive and infinitely many of the guesses made about it by M are wrong, hence f is not inferred by M. If some stage does not terminate then let σ be the finite segment determined up to that point. If M tries to infer the function $\sigma 0^\omega$, then every guess made after seeing σ will diverge on all points past σ, hence M does not infer this function.

$REC \notin BC^a$: Similar to the above construction except that we look for $a + 1$ points to diagonalize on.

$REC \notin [1, m]BC$: Let $M_1, \ldots, M_m$ be any m IIMs. We exhibit a recursive function f that is not BC-inferred by any of them. We construct f in stages. At a stage $s \equiv i \pmod m$, if M_i has not been eliminated, we look for a way to diagonalize against machine M_i (much like the $REC \notin BC$ construction). If we find a way to diagonalize then we do so. If not then we will nonconstructively go to the next stage (meaning that we use the existing finite initial segment as a 'new' starting point in constructing a function); however, we know that *any* recursive function that extends the segment we have at this point cannot be inferred by M_i, so we eliminate M_i from consideration at all later stages.

We end up either constructing a recursive function f not inferred by any of $M_1, \ldots, M_m$ or we end up with a finite segment σ such that any function that extends σ (e.g., $\sigma 0^\omega$) is not inferred by any of $M_1, \ldots, M_m$. In either case we have a recursive function that is not inferred by any of $M_1, \ldots, M_m$.

$REC \notin [1, m]BC^a$: Combine the last two constructions.

$REC \in BC^*$: This proof exploits a 'loophole' in the definition of BC^*. If $\mathcal{S} \in BC^*$ via M then, if $f \in \mathcal{S}$, M can infer f by making guesses $p_1, p_2, p_3, \ldots$ such that, for almost all i, $\varphi_{p_i} =^* f$. Note that the programs could be getting *worse and worse*. (The programs *must* get worse and worse. That is, it is known that if $REC \in BC^*$ via M then there will be recursive functions f

for which the number of errors grows without bound [Che81].)

Here is the inference scheme. Given $\sigma = f(0)f(1)\cdots f(s)$, output a program that does the following. On input x, run each of $\varphi_0, \ldots, \varphi_s$ on $\{0, \ldots, s\}$ for x steps. Let $e \leq s$ be least such that, for all $t \leq s$, $\varphi_{e,x}(t)\downarrow = f(t)$. (For small values of s, such an e may not exist.) If no such e exists, then diverge. If e exists, run the computation of $\varphi_e(x)$ until it halts. (This computation may not halt). If it halts, output $\varphi_e(x)$. For large enough s, x, a correct e will always be found, and this program will output $f(x)$. ∎

The following hierarchy results are not surprising.

Theorem 4.2 *Let $a \in \mathsf{N}$ and $m \geq 1$.*

1. $EX_0 \subset EX_1 \subset EX_2 \subset \cdots \subset EX$ *[CS83].*
2. $EX^0 \subset EX^1 \subset EX^2 \subset \cdots \subset EX^* \subset BC \subset BC^1 \subset BC^2 \cdots \subset BC^*$ *[CS83].*
3. $EX \subset [1,2]EX \subset [1,3]EX \subset \cdots$ *[Smi82].*
4. $[1, m+1]EX \not\subseteq [1,m]BC^a$ *[Smi82]. (This strengthens part 3.)*

Proof sketch:

1) The class in $EX_1 - EX_0$ is

$$\{0^\omega\} \cup \{0^i 1^\omega : i \in \mathsf{N}\}.$$

This can easily be generalized to obtain classes in $EX_{a+1} - EX_a$. The class in $EX - \bigcup_{n=0}^{\infty} EX_n$ is

$$\{f : (\forall^\infty x)[f(x) = 0]\}.$$

2) The class in $EX^{a+1} - EX^a$ is

$$\{f : \varphi_{f(0)} =^{a+1} f\}.$$

The class in $EX^* - \bigcup_{n=0}^{\infty} EX^n$ is

$$\{f : \varphi_{f(0)} =^* f\}.$$

The class in $BC - EX^*$ is

$$\{f : (\forall^\infty x)[\varphi_{f(x)} = f]\}.$$

The class in $BC^{a+1} - BC^a$ is

$$\{f : (\forall^\infty x)[\varphi_{f(x)} =^{a+1} f]\}.$$

The class in $BC^* - \bigcup_{n=0}^{\infty} BC^n$ is REC (see Theorem 4.1).

3,4) The class in $[1,3]EX - [1,2]BC^a$ is

$$\begin{array}{l}\{f : e = \max\{f(0), f(3), f(6), \ldots\} \text{ exists and } f = \varphi_e\} \cup \\ \{f : e = \max\{f(1), f(4), f(7), \ldots\} \text{ exists and } f = \varphi_e\} \cup \\ \{f : e = \max\{f(2), f(5), f(8), \ldots\} \text{ exists and } f = \varphi_e\}.\end{array}$$

This can easily be generalized to obtain classes in $[1, m+1]EX - [1, m]BC^a$. If you just want result 3, then note that the class $\{f : \varphi_{f(0)} =^m f\}$ is in $[1, m+1]EX - [1, m]EX$. This class is in $[1, m+1]EX$ by having IIMs $M_1, \ldots, M_{m+1}$ where machine M_i thinks that $\varphi_{f(0)}$ differs from f on *exactly* $i-1$ points, and tries to patch up the guess $f(0)$ appropriately.

In all three proofs sketched above, the result that the given class of functions is not in the appropriate inference type proceeds by diagonalization. Items 2,3, and 4 use various forms of the Recursion Theorem. ∎

More interesting questions arise if we mix and match parameters. First we will look at mindchanges, anomalies, and teams.

Theorem 4.3 *Let $a, b, c, d \in \mathsf{N}$, $m > 1$, and $n \geq 1$.*

1. *$EX_a^b \subseteq EX_c^d$ iff $a \leq c$ and $b \leq d$ [CS83]. (We can also let a, b, c, or d be $*$, and stipulate that, for all $i \in \mathsf{N}$, $i < *$.)*

2. *Assume $d \geq 2(m-1)$. Then the following are equivalent [FSV89].*

   (a) $[1, m]EX_0^a \subseteq EX_d^c$.

   (b) $c \geq a + \left\lfloor \frac{a(m-1)}{d-(2m-3)} \right\rfloor$.

3. *Assume $c \geq a$. Then the following are equivalent [FSV89]*

   (a) $[1, m]EX_b^a \subseteq EX_d^c$.

   (b)

$$d \geq \left\lfloor \frac{a(m-1)}{c-a+1} \right\rfloor (b+1) + 2mb + \left\lfloor \frac{a}{c-a+1} \right\rfloor b + 2(m-1).$$

4. $[1,m]EX_0^0 \subseteq [1,n]EX_a^0$ *iff* $a \geq \left\lceil \frac{2m}{n} \right\rceil - 2$ *[FSV89].*

Proof sketch:

The inclusion in 1 is obvious. The inclusions in 2 and 3 are proved by complicated simulations where a single machine looks at the $\leq m$ possible guesses and combines them in various ways. The inclusion in 4 is proved similarly, except that n machines look at and combine the guesses. We discuss the non-inclusions.

1) Use classes similar to those used in Theorem 4.2.

2) To prove the necessity of condition b one shows that, if $d < 2(m-1)$ or $c < a + \left\lfloor \frac{a(m-1)}{d-(2m-3)} \right\rfloor$, then the following class is in $[1,m]EX_0^a - EX_d^c$.

$$\{f : \begin{array}{l} f(x) \text{ is odd for no more than } m \text{ distinct values of } x, \text{ and} \\ (\exists x, j)[f(x) = 2j+1 \wedge \varphi_j =^a f]\} \end{array}$$

3,4) Use classes similar to that in the proof of part 2. ∎

By part 1 of the above theorem, there are no tradeoffs between mind-changes and anomalies for EX. The next theorem states that there are such tradeoffs for PEX.

Theorem 4.4 *[CM79, CJM94, GV93] Let* $a, b, c, d \in \mathsf{N}$ *and* $m, n \geq 1$.

1. $PEX_b^a \subseteq PEX_d^c$ *if and only if* $d+1 \geq (b+1)(\left\lfloor \frac{a}{c+1} \right\rfloor + 1)$.
2. $[1,n]PEX_b^a = PEX_{n(b+1)-1}^a$.
3. *There is an algorithm to determine when* $[1,n]PEX_b^a \subseteq [1,m]PEX_d^c$.

We now combine probabilities and teams. We will obtain that every probabilistic inference type is actually equal to some team inference type.

Theorem 4.5 *[Pit89] Let* $0 < p \leq 1$. *Let* n *be the unique positive natural number such that* $\frac{1}{n+1} < p \leq \frac{1}{n}$. *Then* $EX\langle p \rangle = [1,n]EX$ *and* $BC\langle p \rangle = [1,n]BC$. *(See Definition 3.6 for the definition of Probabilistic inference.)*

Proof sketch: We show $[1,n]EX \subseteq EX\langle\frac{1}{n}\rangle$. (A similar proof shows $[1,n]BC \subseteq BC\langle\frac{1}{n}\rangle$.) Assume $\mathcal{S} \in [1,n]EX$ via $\{M_1,\ldots,M_n\}$. The following probabilistic IIM will infer $\mathcal{S}$ with probability $\frac{1}{n}$. First flip an n-sided coin. If it comes up i, then use M_i. The reverse simulation is somewhat complex. ∎

Theorem 4.6 *[Pit89] For all $n \geq 1$, $EX\langle\frac{1}{n}\rangle \subset EX\langle\frac{1}{n+1}\rangle$ and $BC\langle\frac{1}{n}\rangle \subset BC\langle\frac{1}{n+1}\rangle$.*

Proof: This follows from Theorem 4.5 and Theorem 4.2. ∎

For a nice overview of probabilistic learning in the limit, and its relation to teams, see [PS88]. We now combine probabilities, teams, and mindchanges. We only consider 0 mindchanges. Theorems are known about a mindchanges, but the overall picture is less clear.

The first theorem we state is about the case when the probability is over 1/2. In this case virtually everything is known. In addition, the results about probabilistic inference types yield results about team inferences types. The second theorem we state is about the case when the probability is $\leq 1/2$. In this case far less is known.

Theorem 4.7 *Let $k,n,r,s \geq 1$ ($r \leq s$) and $\frac{1}{2} < p \leq 1$.*

1. *If $0 < \frac{r+1}{s+2} < p \leq 1$ then $EX_0\langle p\rangle \subseteq [r,s]EX_0$ [DPVW91].*
2. *If $\frac{n+1}{2n+1} < p \leq \frac{n}{2n-1}$ then (1) $EX_0\langle p\rangle = EX_0\langle\frac{n}{2n-1}\rangle$ [Fre79] and (2) $[n,2n-1]EX_0 \subset [n+1,2n+1]EX_0$ [DPVW91]. (Hence the $EX_0\langle p\rangle$ inference types change at the breakpoints $\frac{2}{3},\frac{3}{5},\frac{4}{7},\ldots$.)*
3. *If $\frac{n+1}{2n+1} < \frac{r}{s} \leq \frac{n}{2n-1}$ then $[r,s]EX_0 = [n,2n-1]EX_0$ [DPVW91].*
4. *If $\frac{r}{s} > \frac{1}{2}$ then $[r,s]EX_0 = [kr,ks]EX_0$ [DPVW91].*
5. *$[n,2n-1]EX_0 = EX_0\langle\frac{n}{2n-1}\rangle$ [DPVW91].*

Item 4 of the last theorem is particularly interesting. It is easy to see that $[r,s]EX_0 \subseteq [kr,ks]EX_0$. ($EX_0$ could be replaced by any inference type.) It is not at all clear that $[kr,ks]EX_0 \subseteq [r,s]EX_0$. This was proved for $\frac{r}{s} > \frac{1}{2}$ by using probabilistic inference types. The next theorem shows that this equality does not hold if $\frac{r}{s} = 1/2$.

Theorem 4.8 *Let $c, d, r, k \geq 1$ with $c \leq d$.*

1. $[1,2]EX_0 \subset [2,4]EX_0$ *[JSV95, Vel89]*
2. $[1,2]EX_0 = [3,6]EX_0 = [5,10]EX_0 = \cdots$, *and* $[2,4]EX_0 = [4,8]EX_0 = [6,12]EX_0 = \cdots$ *[JSV95].*
3. *If* $0 < p_1 \leq \frac{r+1}{kr+1} < p_2 \leq 1$ *then* $EX_0\langle p_2\rangle \subset EX_0\langle p_1\rangle$ *[DPVW91]. Note that these bounds are not necessarily best possible.*
4. *Let p, c, d be such that $\frac{24}{49} < p = \frac{c}{d} < \frac{1}{2}$. Then (1) $EX\langle p\rangle = [2,4]EX_0$, (2) $[c,d]EX_0 = [2,4]EX_0$, and (3) $[2,4]EX_0 \subset [24,49]EX_0$ [DKV95].*

Proof sketch: One feels a moral obligation to comment on the numbers 24 and 49. Efim Kinber commented (jokingly) that they are 'nonrecursive numbers' meaning only that they seem unnatural. The authors of the paper obtained them by trial and error, with the help of a computer program. ∎

The fraction $\frac{24}{49}$ is called a *breakpoint for* EX_0. We leave it to the reader to define this term rigorously. A few of the breakpoints for $EX_0\langle p\rangle$ less than $\frac{24}{49}$ are known, but each one requires a new proof and no pattern is known. If only total functions can be output then the following stunning result is known.

Theorem 4.9 *[Amb96]*

1. *The set of breakpoints for PEX_0 has order type ϵ_0. (ϵ_0 is the ordinal that is the limit of the sequence $x_0 = \omega$, $x_{i+1} = \omega^{x_i}$.)*
2. *The problem of, given $p, q \in [0,1]$, determining if $PEX_0\langle p\rangle \subset PEX_0\langle q\rangle$ is decidable.*

(For all $p \in [0,1]$, $PEX_0\langle p\rangle$ is called $PFIN\langle p\rangle$ in [Amb96].)

Open Problem 4.10 Find an algorithm that will, given $a, b, c, d \geq 1$ (where $a \leq b$ and $c \leq d$), determine if $[a,b]EX_0 \subseteq [c,d]EX_0$. The problem for $[a,b]PEX_0 \subseteq [c,d]PEX_0$ has been solved [Kum94].

5 Query Inference: Asking About the Function

The scenario so far has been (roughly) that of a learner receiving $f(0)$, $f(1)$, $f(2)$, ... and trying to infer f. What if the learner could also ask questions about f?

If I said to you '$2, 4, 6, 8$' and you got to ask a question about the function I am presenting, then you might ask '$(\forall x)[f(x) = 2x]$?' If I answer YES then you could, with confidence, claim that $f(x) = 2x$ was the function. If I said NO then you could wait for more data. Say I extend the data to $2, 4, 6, 8, 8, 8$. Then you could ask '$(\forall x)[x \geq 4 \Rightarrow f(x) = 8]$?' If I answer YES then you could output that the function is

$$f(x) = \begin{cases} 2x & \text{if } x = 1, 2, 3, 4; \\ 8 & \text{otherwise.} \end{cases}$$

Note that you are fully confident of your guess.

We will formalize what it means to be able to ask questions about the function. We picture a teacher who is trying to teach you a function f by answering certain questions about f. In a later section (Section 7) we will be asking questions to a fellow student who may be very bright but does not know anything about f.

If you can ask questions then can you learn more than otherwise? Can you learn REC? This may depend on both the kind of questions you are allowed to ask and the kind of learning you wish to do.

We will define a *query inference machine* as one that makes queries in some language about the function to be learned. We first define carefully what a query is. We define queries about *functions*; however the analogous notions for *sets* are easy to define.

Definition 5.1 A *query language* consists of the usual logical symbols (and equality), symbols for number variables and set variables, symbols for every element of N, symbols for some functions and relations on N, and a special symbol $\mathcal{F}$ denoting a function we wish to learn. A query language is denoted by the symbols for the functions and relations in the language. We will use a superscript 2 to indicate that we allow quantification over set variables. For example, we refer to 'the query language $[+, <]$' or 'the query language

$[Succ, <]^2$.' A well-formed formula over a query language L is defined in the usual way.

Convention 5.2 Small letters are used for number variables, which range over N. Capital letters are used for set variables, which range over subsets of N. In all languages considered here, the symbols in the language represent recursive operations.

Definition 5.3 Let L be a query language. A *query over* L is a formula $\phi(\mathcal{F})$ such that the following hold.

1. $\phi(\mathcal{F})$ uses symbols from L.
2. $\mathcal{F}$ is the special function symbol.
3. $\phi(\mathcal{F})$ has no free variables aside from $\mathcal{F}$.

We think of a query $\phi(\mathcal{F})$ as making a query about an as yet unspecified function $f : \mathsf{N} \to \mathsf{N}$. For such an f, $\phi(f)$ will be either true or false.

Notation 5.4 Let $b \geq 2$. We will be using the following symbols in some of our query languages

1. $Succ$ stands for the successor function $Succ(x) = x + 1$.
2. POW_b is the unary predicate that determines if a number is in the set $\{b^n : n \in \mathsf{N}\}$.
3. POLY_b is the unary predicate that determines if a number is in the set $\{n^b : n \in \mathsf{N}\}$.
4. FAC is the unary predicate that determines if a number is in the set $\{n! : n \in \mathsf{N}\}$.
5. If MOD_b is one of the auxiliary symbols in L this means that L has b unary predicates that determine membership in the following b sets. $U_i = \{n : n \equiv i \pmod{b}\}$ for $0 \leq i \leq b-1$.

Definition 5.5 A query inference machine (QIM) is a total Turing machine that can make queries about the recursive function f in a particular query language (and magically get the answers). Note that a QIM gets all of its information from making queries and does not see any data; however it can request whatever data it wants (e.g., the QIM can ask '$f(0) = 0$?', '$f(0) = 1$?',... until a YES is encountered). During every stage it makes a query, gets an answer, and then guesses an index for f. (See Figure 2.)

Note 5.6 We will often restrict to the case where we are learning a class of recursive *sets* instead of functions. In this case our query language has a special set symbol X, and a symbol $\in$, instead of a special function symbol $\mathcal{F}$.

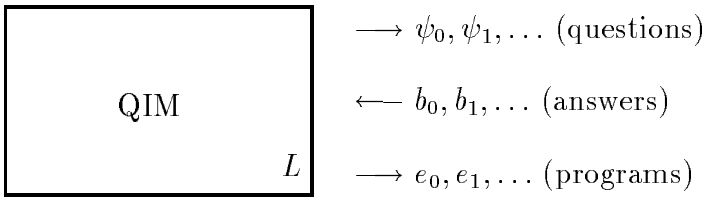

Figure 2

Definition 5.7 $QEX[L]$ is the collection of all $\mathcal{S} \subseteq REC$ that are inferred by some QIM that makes queries in the query language L. For $i \in \mathsf{N}$, $Q_iEX[L]$ is the collection of all $\mathcal{S} \subseteq REC$ which can be inferred in the limit by a QIM that makes queries in the query language L, where the queries are well formed formulas over L that are expressed in prenex normal form and the formulas are restricted to at most i 'blocks' of quantifiers. Each block is a string of either finitely many existential quantifiers or finitely many universal quantifiers, and adjacent blocks have opposite quantifier types. (Note that queries made in $Q_0EX[L]$ are quantifier free, and, for all $i > 0$, queries made in $Q_iEX[L]$ have at most $i - 1$ *alternations* of quantifiers.)

Definition 5.8 If $\mathcal{I}$ is any of the inference types from Definition 3.6 then the corresponding query inference types $Q\mathcal{I}[L]$ and $Q_i\mathcal{I}[L]$ can easily be defined. We will denote team-query inference types by (say) $[c,d]QEX$ rather than $Q[c,d]EX$.

6 Results about Query Inference

6.1 Trying to Learn REC

The question of whether you can infer REC using queries to L has been answered for several L.

We consider learning REC with the query language $[+, \times]$. Under certain conditions REC can be learned with this query language. In the next two theorems, we investigate how much power suffices—and how much is needed—to infer REC. We look at both the number of alternations of quantifiers in the queries and the number of mindchanges the QIM is allowed to make.

Theorem 6.1 ([GS92]) $REC \in Q_1EX[+, \times]$. $REC \in Q_2EX_0[+, \times]$.

Proof sketch: Both proofs use the following theorem (proved in [Mat70], but builds on work from [DPR61]): If A is any r.e. set, then there exists a polynomial $p(x, x_1, \ldots, x_k)$ (with integer coefficients) such that

$$A = \{x : (\exists x_1, \ldots, x_k)[p(x, x_1, \ldots, x_k) = 0]\}.$$

This theorem was an outgrowth of work done on Hilbert's 10th problem (see [Mat93] for more on Hilbert's 10th problem).

As we will see in Section 7, REC can be inferred by using an ordinary IIM together with queries of the form '$y \in K$?' (for $y \in \mathbb{N}$). Letting $A = K$ above, we see that we can make existential queries over $[+, \times]$ in place of making queries to K; hence $REC \in Q_1EX[+, \times]$. The proof that $REC \in Q_2EX_0[+, \times]$ is left to the reader. ∎

Theorem 6.2 ([GPS92]) *If every symbol in L represents a recursive relation or function then $EX_{i+1} \not\subseteq Q_1EX_i[L]$. In particular, for any i, $REC \notin Q_1EX_i[L]$.*

Proof sketch: We describe the class used that is in EX_1 but not in $Q_1EX_0[+, \times]$. A function f *has a cycle of length d* if there exists a such that $f(a) = a+1$, $f(a+1) = a+2$, ..., $f(a+d-1) = a+d$ and $f(a+d) = a$. Let $\mathcal{S}$ be the class

$\{f : \varphi_{f(0)} = f$ and f has no cycles $\} \cup$
$\{f : (\exists d)[f$ has exactly one cycle, its length is d, and $f = \varphi_d]\}$.

The intuitive reason that $\mathcal{S}$ is not learnable in $Q_1EX_0[+,\times]$ is that, while you can ask 'does there exist a cycle of length d?' it is impossible to ask 'does there exist a cycle?' ∎

We now consider the language $[Succ, <]^2$. There are two reasons for considering this language: (1) queries in $[Succ, <]^2$ can be interpreted as Büchi automata (to be defined later), hence we have the mathematical tools to prove sophisticated theorems, and (2) other languages can be 'reduced' to this one (see [GH95] or Definition 6.11), hence results about this language imply results about other languages.

For this study we will consider inferring sets instead of functions. This will not affect any of the results because if (say) $RECSET \notin QEX[Succ, <]^2$ then $REC \notin QEX[Succ, <]^2$.

Definition 6.3 A *Büchi Automaton* [Büc62] is a nondeterministic finite automaton $\mathcal{A} = \langle Q, \Sigma, \Delta, s, F\rangle$ where Q is the set of *states*, Σ is the *alphabet*, Δ maps $Q \times \Sigma$ to 2^Q, $s \in Q$ (s is the *start state*), and $F \subseteq Q$ (F is the set of *accepting states*). The reason we have a new name for a familiar device is that Büchi Automata are used on elements of Σ^ω (i.e., on *infinite* strings of elements of Σ). Let $\vec{x} \in \Sigma^\omega$. A *run of $\mathcal{A}$ on $\vec{x}$* is a sequence $\vec{q} \in Q^\omega$ such that $\vec{q}[0] = s$ and $(\forall i)[\vec{q}[i+1] \in \Delta(\vec{q}[i], \vec{x}[i])]$. *$\mathcal{A}$ accepts $\vec{x}$* if there exists a run $\vec{q}$ such that $(\exists^\infty i)[\vec{q}[i] \in F]$. *$\mathcal{A}$ accepts a set $A \subseteq \Sigma^\omega$* if $(\forall \vec{x} \in \Sigma^\omega)[\vec{x} \in A$ iff $\mathcal{A}$ accepts $\vec{x}$ $]$. If $A \subseteq \Sigma^\omega$ and there exists a Büchi automaton that accepts all the strings in A and no others, then A is called *ω-regular*.

By representing sets of natural numbers via their characteristic sequences (which are elements of $\{0,1\}^\omega$), we can think of an ω-regular set over the alphabet $\Sigma = \{0,1\}$ as a subset of $\mathcal{P}(\mathbb{N})$ (the power set of $\mathbb{N}$). Thus we can represent a k-tuple of sets of natural numbers via an element of $(\{0,1\}^k)^\omega$, and an ω-regular set over the alphabet $\Sigma = \{0,1\}^k$ as being a subset of $\mathcal{P}(\mathbb{N}) \times \cdots \times \mathcal{P}(\mathbb{N})$ (there are k copies of $\mathcal{P}(\mathbb{N})$). We denote the input to a Büchi automaton over the alphabet $\Sigma = \{0,1\}^k$ by $(A_1, \ldots, A_k)$ where A_i is

the set of natural numbers whose characteristic sequence is the projection of the input onto the ith coordinate. By using singleton sets, we can also code numbers. The following theorems from the literature link Büchi automata with formulas over $[Succ, <]^2$.

Lemma 6.4 ([Büc60, Büc62, Cho74]) *If $\phi(x_1, \ldots, x_{k_1}, X_1, \ldots, X_{k_2})$ is a formula over $[Succ, <]^2$ then the set*

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

is ω-regular. Furthermore, there is a recursive procedure to transform any formula into the appropriate Büchi automaton. (One can also, given an automaton, find an equivalent formula. We do not use this.)

Proof: This proof is by induction on the formation of a formula. Atomic formulas are easy. The $\wedge$ and $\vee$ of formulas is handled by using closure of ω-regular sets under intersection and union (proved with cross-product constructions). Existential quantifiers are handled easily, since Büchi automata are nondeterministic. The only real hard step is showing that ω-regular sets are closed under complementation. This was first shown in [Büc62]. See [Cho74] for a good exposition and see [Saf88] for a more efficient proof. ∎

Lemma 6.5 *If $q(X)$ is a query in $[Succ, <]^2$ then the set $\{A : q(A)\}$ is ω-regular. Furthermore, there is a recursive procedure to transform any query into the appropriate Büchi automaton.*

Note 6.6 Here are two facts of interest that we will not be using. (1) Given a Büchi automaton, one can effectively construct a corresponding formula in $[Succ, <]^2$ (in fact, with only four quantifiers). (2) It is decidable to test whether there is some set (of infinite strings) which is accepted by a given Büchi automaton. This, together with Lemma 6.4, enables one to prove that the theory of one successor (which also includes $<$), denoted S1S, is decidable. This was Büchi's original motivation.

Definition 6.7 If $\mathcal{A}$ is a Büchi automaton then $L(\mathcal{A})$ is the set of (infinite) strings that are accepted by it.

Theorem 6.8 ([GH95, GS92]) *Let $a \in \mathtt{N}$ and $d \geq 1$. Then*

1. $REC \notin QEX[Succ, <]^2$.
2. $REC \notin QEX^a[Succ, <]^2$.
3. $REC \notin [1, d]QBC^a[Succ, <]^2$.

Proof sketch:

The proof that $REC \notin QEX[Succ, <]^2$ is analogous to the proof that $REC \notin EX$, except that it is much harder. Let M be a $QEX[Succ, <]^2$ machine. We construct a recursive, 0,1-valued function f such that f is not inferred by M. (Note that f is equivalent to the characteristic sequence of some set of natural numbers.) We build f in stages. At stage s we have an initial segment f_s of f ($f_s \in \{0,1\}^*$) and a Büchi automaton $\mathcal{A}_s$. We intend that the final function f is accepted by $\mathcal{A}_s$. We also need that, for all s, $L(\mathcal{A}_s)$ is uncountable.

During stage s we look for (1) a convergence of the current guess on a point x (chosen so that both $L(\mathcal{A}_s) \cap \{B : x \in B\}$ and $L(\mathcal{A}_s) \cap \{B : x \notin B\}$ are both uncountable) so that we can diagonalize, and (2) a finite sequence of queries—and accompanying answers—that causes a mindchange. At the same time we build a default function f'. Either (1) convergence is found, so we can diagonalize, throw out the default function, and proceed to the next stage, or (2) a mindchange is found, so we answer the queries to go that way, and proceed to the next stage, or (3) neither convergence nor a mindchange is found, in which case f' is total recursive and not inferred by M.

To obtain the results for QEX^a and $[1, d]QBC^a$, we combine the proof above with the proofs that $REC \notin EX^a$ and $REC \notin [1, d]BC^a$. ∎

We now define a notion of reduction that will enable us to obtain the result $REC \notin QEX[L]$ for many L. For some of these L (e.g., $L = [+, <]$) there were already proofs in the literature of $REC \notin QEX[L]$; however, the new proofs are easier. For other L (e.g., $L = [Succ, <, FAC]^2$) the results are new.

Definition 6.9 If E is any set, then E^* is the set of all finite sequences over E.

Notation 6.10 Let E be an infinite subset of N, and f be a bijection between E and N. We extend the definition of f to $\mathcal{P}(E)$ (the power set of E) and E^* by $f(A) = \{f(n) : n \in A\}$ and $f(n_1, \ldots, n_a) = (f(n_1), \ldots, f(n_a))$, respectively. Note that f is a bijection between E and N, and that the extensions of f to $\mathcal{P}(E)$ and E^* are bijections between $\mathcal{P}(E)$ and $\mathcal{P}(\mathsf{N})$, and E^* and N^*, respectively). We will use f to denote all three of these functions. The meaning will be clear from context.

Definition 6.11 [GH95] Let L, $L^\$$ be query languages and E be an infinite recursive subset of N. Let f be a recursive bijection from E to N. Formally everything in this definition is parameterized by E and f; however we will not make this explicit. *L is reducible to $L^\$$*, written $L \leq L^\$$, if there exists a recursive function with the following properties.

1. The input is a query $\psi(X)$ over L and the output is a query $\psi^\$(X)$ over $L^\$$. (This is called *the domain condition.*)
2. $(\forall A \subseteq E)[\psi(A)$ iff $\psi^\$(f(A))]$. (This is called *the equivalence condition.*)

(What we denote by $\leq$ was denoted by $\leq_{\mathsf{N}}$ in [GH95].)

The following lemma has a straightforward but tedious proof. The hard part of using reductions was coming up with the right definition.

Lemma 6.12 *[GH95] If $L \leq L^\$$ with parameters (E, f) then the following hold.*

1. *Let $\mathcal{I}$ be any of the inference types discussed in Section 3. Let $Q\mathcal{I}$ be the corresponding query notion. For all $\mathcal{A} \subseteq RECSET \cap \{X : X \subseteq E\}$ the following hold.*
   (a) *$\mathcal{A} \in \mathcal{I}$ iff $f(\mathcal{A}) \in \mathcal{I}$.*
   (b) *$\mathcal{A} \in Q\mathcal{I}[L] \Rightarrow f(\mathcal{A}) \in Q\mathcal{I}[L^\$]$.*
   (c) *$REC \in Q\mathcal{I}[L] \Rightarrow REC \in Q\mathcal{I}[L^\$]$.*
2. *If $U, V \subseteq E$ then $[U =^a V$ iff $f(U) =^a f(V)]$. (This is trivial, and is independent of the condition that $L \leq L^\$$.)*

Lemma 6.13 ([GH95]) *Let $b \geq 2$.*

1. $[+, <] \leq [Succ, <]^2$.
2. $[+, <, \text{POW}_b, \text{MOD}_b] \leq [Succ, <]^2$.

Proof sketch: The proof is similar to the proof that Presburger arithmetic is decidable by reduction to the weak second-order theory with one successor (often called WS1S) [Rab77]. ∎

From Lemmas 6.12 and 6.13 we easily obtain the following.

Theorem 6.14 ([GH95]) *Let $a, b, d \geq 2$.*

1. $REC \notin [1, d]QBC^a[+, <]$.
2. $REC \notin [1, d]QBC^a[+, <, \text{POW}_b, \text{MOD}_b]$.

The result $REC \notin QEX[+, <]$ was initially shown [GPS92] using a device called k-good sets.

6.2 Comparing Classes

Here are some interesting theorems comparing the power of IIMs to that of QIMs.

Theorem 6.15 *Let $a, d, n \in \mathsf{N}$ and $b \geq 2$. The following hold.*

1. *For any query language L, $EX_{n+1} \not\subseteq Q_1EX_n^a[L]$ [GH95, GPS92].*
2. *For any query language L, $Q_1EX_0[L] \subseteq EX$ [GS92]. (This can be generalized to $Q_{d+1}EX_n[L] \subseteq [1, n+1]Q_dEX[L]$ [GKP+95].)*
3. *Let L be any of $[Succ, <]^2$, $[Succ, <, \text{FAC}]^2$, $[Succ, <, \text{POLY}_b]$, $[Succ, <, \text{POW}_b]$, and $[+, <, POW_b, MOD_b]$. Then*
   (a) *$Q_1EX_n[L] \subset [1, n+1]EX$ [GKP+95, GH95].*
   (b) *$d \geq 1 \Rightarrow EX_0^* \not\subseteq [1, d]QEX^a[L]$ [GH95].*

Proof sketch:

1) See Theorem 6.2.

2) We show that $Q_1EX_0[L] \subseteq EX$. Whenever a query is made, assume that the answer is NO, but always be on the lookout for evidence that the answer is YES. If such evidence is found, then restart the simulation with that information (note that this may affect the set of questions the query machine asks). Note that every query can be put into existential form, so if the answer to a query really is YES, this fact will eventually be discovered. Since no mindchanges can be made by this QIM, there exists some point (which will not be known by the simulator) at which all the queries made have been given the correct answer, a guess has been made, and this guess is correct.

3) For all of these we first prove the result with $L = [Succ, <]^2$ and then use reductions like the one in Definition 6.11. ∎

Here are several examples of when a query machine with a restriction on its behavior (e.g., at most one mindchange) can still do things that a standard machine cannot do.

Theorem 6.16 *Let $a \in \mathbb{N}$ and $c \geq 1$.*

1. *For any L, $Q_1EX_1[L] \not\subseteq EX$ [GS92].*
2. *$Q_2EX_0[<] \not\subseteq EX$ [GS92].*
3. *$Q_1EX_c[Succ] \not\subseteq [1,c]BC^a$ [GKP+95].*

Proof sketch:

1) Let

$$\mathcal{S} = \{f : \varphi_{f(0)} = f\} \cup \{f : (\forall^\infty x)[f(x) = 0]\}.$$

$\mathcal{S} \notin EX$ by an easy diagonalization that uses the Recursion Theorem. $\mathcal{S} \in Q_1EX_1[L]$ as follows. Output the value $f(0)$ as the first guess. Then determine if f is almost always 0 by asking:

$$\begin{array}{ll} (\forall x) & [x \neq 0 \Rightarrow \mathcal{F}(x) = 0] \\ (\forall x) & [x \neq 0 \wedge x \neq 1 \Rightarrow \mathcal{F}(x) = 0] \\ (\forall x) & [x \neq 0 \wedge x \neq 1 \wedge x \neq 2 \Rightarrow \mathcal{F}(x) = 0] \\ & \vdots \end{array}$$

until (if ever) a YES answer is encountered. If a YES answer is never encountered, then $f(0)$, the QIM's first conjecture, is correct. If a YES answer is encountered, then output the appropriate program.

2) The class $\mathcal{S}$ defined in part 1 is in $Q_2EX_0[<]$. First ask whether $(\exists x)(\forall y)[y > x \Rightarrow \mathcal{F}(y) = 0]$. If the answer is NO, then output $f(0)$. If the answer is YES, then ask a series of questions to find out at which point the function is always 0. Output the appropriate program. By part 1, $\mathcal{S} \notin EX$. Hence $Q_2EX_0[<] \not\subseteq EX$.

3) We consider the case of $c = 3$. The general case is similar. For $0 \leq i \leq 3$ let f_i be defined as $f_i(x) = f(4x + i)$. Hence f_i captures all the information of f restricted to $\{x : x \equiv i \pmod 4\}$.

Let $\mathcal{S}$ be the union of the following four classes, $\mathcal{S}_1, \mathcal{S}_2$, $\mathcal{S}_3$, and $\mathcal{S}_4$.

$f \in \mathcal{S}_1$ if

1. There exists e_1 such that $(\forall x)[f_0(x) = e_1]$ and $f = \varphi_{e_1}$.
2. There is no y such that $(\forall^\infty x)[f_1(x) = y]$.
3. There is no y such that $(\forall^\infty x)[f_2(x) = y]$.
4. There is no y such that $(\forall^\infty x)[f_3(x) = y]$.
5. There are no $y \in \mathsf{N}$ and $j \geq 1$ such that $f_j(y) = e_1$.

(Note that all these conditions must hold for f to be in $\mathcal{S}_1$. The same will be the case in the definitions of $\mathcal{S}_2$, $\mathcal{S}_3$, and $\mathcal{S}_4$ below.)

$f \in \mathcal{S}_2$ if

1. There exists e_1 such that $(\forall x)[f_0(x) = e_1]$
2. There exists e_2 such that $(\forall^\infty x)[f_1(x) = e_2]$ and $f = \varphi_{e_2}$.
3. There is no y such that $(\forall^\infty x)[f_2(x) = y]$.
4. There is no y such that $(\forall^\infty x)[f_3(x) = y]$.
5. There are no $y \in \mathsf{N}$ and $j \geq 2$ such that $f_j(y) = e_2$.

$f \in \mathcal{S}_3$ if

1. There exists e_1 such that $(\forall x)[f_0(x) = e_1]$
2. There exists e_2 such that $(\forall^\infty x)[f_1(x) = e_2]$.
3. There exists e_3 such that $(\forall^\infty x)[f_2(x) = e_3]$ and $f = \varphi_{e_3}$.
4. There is no y such that $(\forall^\infty x)[f_3(x) = y]$.
5. There is no $y \in \mathsf{N}$ such that $f_3(y) = e_3$.

$f \in \mathcal{S}_4$ if

1. There exists e_1 such that $(\forall x)[f_0(x) = e_1]$
2. There exists e_2 such that $(\forall^\infty x)[f_1(x) = e_2]$.
3. There exists e_3 such that $(\forall^\infty x)[f_2(x) = e_3]$ and $(\forall^\infty x)[f_3(x) = e_3]$.
4.

We show $\mathcal{S} \in Q_1EX_3[Succ]$. At all times we will have an index for a candidate that we will keep on outputting until a better candidate is discovered.

The value of e_1 can easily be found. Output this value. To find e_2 (if it exists), ask the following question for all values of a and b:

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

If a "YES" answer is received, then $e_2 = b$. Output e_2. By a similar process, find e_3 (if it exists). If e_3 is found we do not know if we are in $\mathcal{S}_3$ or $\mathcal{S}_4$. Output e_3. To discover if we are actually in $\mathcal{S}_4$ ask the following question for all values of a:

$$\forall y[(y \neq 0 \text{ and } y \neq 1 \text{ and } \ldots \text{ and } y \neq a) \Rightarrow (\mathcal{F}(y) = e_3 \Rightarrow \mathcal{F}(y+1) = e_3)]?$$

If a "YES" is ever encountered then we know the function is in $\mathcal{S}_4$. From the information gathered so far, and a few more questions, we can determine a program for f.

To show that $\mathcal{S} \notin [1,3]BC^a$, we use the Operator Recursion Theorem [Cas74] and diagonalization. For an easier use of the Operator Recursion Theorem in inductive inference, see [CS83].

∎

Open Problem 6.17 We conjecture that for $L \leq [Succ, <]^2$ the following hold.

1. $QEX^1[L] \subset QEX^2[L] \subset QEX^3[L] \cdots$.
2. $QBC^1[L] \subset QBC^2[L] \subset QBC^3[L] \cdots$.
3. $QEX[L] \subset [1,2]QEX[L] \subset [1,3]QEX[L] \cdots$.
4. $QBC[L] \subset [1,2]QBC[L] \subset [1,3]QBC[L] \cdots$.

More generally we conjecture that whatever relations hold between $[c,d]EX^a$ and $[c',d']EX^{a'}$ also hold between the corresponding query inference types. We do not actually believe this conjecture; however, we think it would be of interest to see which theorems for EX hold here. The same goes for BC. The classes in Theorem 4.2 may be good candidates to help solve these conjectures.

Open Problem 6.18 Do more alternations of quantifiers help? Conjectures:

1. $Q_1EX[Succ, <] \subset Q_2EX[Succ, <] \subset Q_3EX[Succ, <] \cdots$.
2. $Q_1EX[+, <] \subset Q_2EX[+, <] \subset Q_3EX[+, <] \cdots$.

It is known that the $Q_iEX[Succ, <]^2$ hierarchy collapses, since every query is equivalent to a Büchi automaton, which is in turn equivalent to a (second-order) formula with four alternations of quantifiers. It is also known that, for all $i \geq 1$, $Q_iEX[+, \times] = Q_{i+1}EX[+, \times]$ since $REC \in Q_1EX[+, \times]$. Therefore, if the conjecture is true, the proof will have to use particular properties of $[Succ, <]$ and $[+, <]$.

7 Query Inference: Asking an Oracle

Our first scenario was (roughly) that of a learner receiving $f(0), f(1), f(2), \ldots$ and trying to infer f. Our second scenario was (roughly) that of a learner asking questions about f. Our third scenario is that of a learner receiving $f(0), f(1), f(2), \ldots$ and also querying an oracle A. This oracle is independent of f. Our mental picture is that the learner is asking a fellow student for help. The student does not know f but might know information that is useful.

Definition 7.1 An *oracle inductive inference machine* (OIIM) $M^{()}$ is an Oracle Turing machine $M^{()}$ such that, for all $\sigma \in \{0,1\}^*$ and all sets A of natural numbers, $M^A(\sigma)\downarrow$. We interpret M^A to be trying to learn a recursive function f similarly to our interpretation of an IIM M trying to learn a recursive function f. We define M^A $EX[A]$-*identifies* f similarly to our definition of M EX-identifies f. For any inference type $\mathcal{I}$, the notation $\mathcal{I}[A]$ can be defined.

Definition 7.2 Let $\mathcal{S}$ be a class of recursive functions. $\mathcal{S} \in EX[A]$ is defined similarly to $\mathcal{S} \in EX$. If $\mathcal{I}$ is any inference type, then $\mathcal{S} \in \mathcal{I}[A]$ is defined similarly to $\mathcal{S} \in \mathcal{I}$.

Note that in the definition of $EX[A]$ (and the other inference types) we are inferring indices for recursive functions, *not* indices for recursive-in-A functions.

Example 7.3 $REC \in EX[K]$. Upon seeing initial segment σ, output the least e such that $(\forall x < |\sigma|)[\varphi_e(x)\downarrow = \sigma(x)]$. Note that even after the correct index is found infinitely many queries to K are made to keep verifying that the index is correct.

Note 7.4 Using $REC \in EX[K]$ we can give an alternative proof that $REC \in BC^*$ (Theorem 4.1). Let $M^{()}$ be the OIIM such that M^K infers REC. Let N be the IIM that operates as follows: $\varphi_{N(\sigma)}(s) = \varphi_{M^{K_s}(\sigma)}(s)$. It is easy to show that N BC^*-infers REC.

Example 7.5 Recall the notation of A' from Notation 2.5. Let

$$\mathcal{S} = \{f : f(0) \in A' \wedge \varphi_{f(1)} = f\} \cup \{f : f(0) \notin A' \wedge f =^* \lambda x[0]\}.$$

Note that $\mathcal{S} \in EX[A]$ by using a recursive-in-A approximation to A', which exists by the Limit Lemma (see [Soa87, p. 57]).

Definition 7.6 Let $\mathcal{S}$ be a class of recursive functions. $\mathcal{S} \in EX[A*]$ if there exists an OIIM $M^{()}$ such that (1) $\mathcal{S}$ is $EX[A]$-identified by M^A, and (2) for every $f \in \mathcal{S}$, during the inference of f by M^A, only finitely many queries to A are made. $BC[A*]$ is defined similarly. (The notion of $EX[A[i]]$, where at most i queries to A are made, was studied in [FJG+94]. We will not discuss it here.)

Example 7.7 $REC \in EX[TOT*]$. Upon seeing $f(0), \ldots, f(s)$, output the least index i such that $i \in TOT$ and, for all $t \leq s$, $\varphi_i(t) \downarrow = f(t)$. Note that if j is least such that φ_j computes f, then at most $j+1$ queries to TOT will be made.

Example 7.8 Let

$$T = \{f : (\exists i)[(f(0), f(1), \ldots, f(i) \in A) \wedge (f(i+1) \notin A) \wedge (\varphi_{f(i)} = f)\}.$$

It is easy to see that $T \in EX[A*]$.

Definition 7.9 $A \leq_{\mathrm{i}} B$ if $EX[A] \subseteq EX[B]$ (the 'i' stands for 'inference'). $A \equiv_{\mathrm{i}} B$ if $EX[A] = EX[B]$. An *EX-degree* is an equivalence class under $\equiv_{\mathrm{i}}$. $A \leq_{\mathrm{i}}^* B$ if $EX[A*] \subseteq EX[B*]$. $A \equiv_{\mathrm{i}}^* B$ if $EX[A*] = EX[B*]$. An *$EX*$-degree* is an equivalence class under $\equiv_{\mathrm{i}}^*$. The BC-degrees and $BC*$-degrees are defined similarly. More generally, these definitions would make sense for any of the inference types defined in Definition 3.6. These degree structures are spoken of informally as the *degrees of inferability*.

Definition 7.10 An EX-degree is *trivial* if for every A in that degree, $EX[A] = EX$. An EX-degree is *omniscient* if for every A in that degree $REC \in EX[A]$. The notions trivial and omniscient can be defined for other types of degrees of inferability.

7.1 Extremes in the Degrees of Inferability

Which oracles help you learn REC (e.g., $REC \in EX[A]$)? Which oracles do not help at all (e.g., $EX[A] = EX$)? The answers depend on your model of learning. The following table tells (almost) all. TRIV means that the oracle does not help at all. $*$-TRIV means that if we only allow finitely many questions, then the oracle does not help at all. OMNI means that the oracle helps to learn REC. $*$-OMNI means that the oracle helps to learn REC and that the OIIM makes only finitely many queries to the oracle. We will need another definition.

Definition 7.11 A set G is *i-generic* if for every Σ_i set W (of elements of $\{0,1\}^*$) either (1) $(\exists \sigma \preceq G)[\sigma \in W]$, in which case we say that G *meets* W, or (2) $(\exists \sigma \preceq G)[(\forall \tau \succeq \sigma)[\tau \notin W]]$. (For more on 1-generic sets see [Joc80].)

Definition 7.12 If A is a set of natural numbers, then $\mathcal{G}(A)$ means that one of the following holds.

1. A is recursive.
2. $A \leq_{\mathrm{T}} K$, and A is in a 1-generic degree.

In Table 7.1, $U_1, \ldots, U_5$ are open questions. U_4 and U_5 may depend on the parameters a, b, n. The results for EX being omniscient are in [AB91, KS96]. The results for EX being trivial are in [SS91, KS96]. All the rest are in [FJG+94]. We know more than what is in the table [FJG+94].

1. If $A \leq_{\mathrm{T}} K$ or A is in a hyperimmune-free degree then $U_1(A)$ holds.
2. There are low sets A such that $U_2(A)$ and $U_5(A)$ hold.
3. For all sets X there exists a set A such that $X \leq_{\mathrm{T}} A''$ but neither $U_2(A)$ nor $U_5(A)$ holds.
4. If A is r.e. then the following are equivalent: (1) $U_2(A)$, (2) $U_4(A)$, (3) $\emptyset'' \leq_{\mathrm{T}} A'$.
5. For all A, $U_3(A) \Rightarrow U_4(A) \Rightarrow \mathcal{G}(A)$.

	TRIV	$*$-TRIV	OMNI	$*$-OMNI
PEX	$U_1(A)$	$U_1(A)$	$\emptyset'' \leq_{\mathrm{T}} A \oplus K$	$\emptyset'' \leq_{\mathrm{T}} A \oplus K$
EX_m	$A \equiv_{\mathrm{T}} \emptyset$	$A \equiv_{\mathrm{T}} \emptyset$	Never	Never
EX	$\mathcal{G}(A)$	$A \leq_{\mathrm{T}} K$	$\emptyset'' \leq_{\mathrm{T}} A'$	$\emptyset'' \leq_{\mathrm{T}} A \oplus K$
EX^n	$\mathcal{G}(A)$	$A \leq_{\mathrm{T}} K$	$\emptyset'' \leq_{\mathrm{T}} A'$	$\emptyset'' \leq_{\mathrm{T}} A \oplus K$
EX^*	$\mathcal{G}(A)$	$A \leq_{\mathrm{T}} K$	$\emptyset'' \leq_{\mathrm{T}} A'$	$\emptyset'' \leq_{\mathrm{T}} A \oplus K$
BC	$\mathcal{G}(A)$	$A \leq_{\mathrm{T}} K$	$U_2(A)$	$\emptyset'' \leq_{\mathrm{T}} A \oplus K$
BC^n	$U_3(A)$	$A \leq_{\mathrm{T}} K$	$U_2(A)$	$\emptyset'' \leq_{\mathrm{T}} A \oplus K$
BC^*	Always	Always	Always	Always
$[a,b]EX_m^n$	$A \equiv_{\mathrm{T}} \emptyset$	$A \equiv_{\mathrm{T}} \emptyset$	Never	Never
$[a,b]EX_m^*$	$A \equiv_{\mathrm{T}} \emptyset$	$A \equiv_{\mathrm{T}} \emptyset$	Never	Never
$[a,b]EX$	$\mathcal{G}(A)$	$A \leq_{\mathrm{T}} K$	$\emptyset'' \leq_{\mathrm{T}} A'$	$\emptyset'' \leq_{\mathrm{T}} A \oplus K$
$[a,b]EX^n$	$\mathcal{G}(A)$	$A \leq_{\mathrm{T}} K$	$\emptyset'' \leq_{\mathrm{T}} A'$	$\emptyset'' \leq_{\mathrm{T}} A \oplus K$
$[a,b]EX^*$	$\mathcal{G}(A)$	$A \leq_{\mathrm{T}} K$	$\emptyset'' \leq_{\mathrm{T}} A'$	$\emptyset'' \leq_{\mathrm{T}} A \oplus K$
$[a,b]BC$	$\mathcal{G}(A)$	$A \leq_{\mathrm{T}} K$	$U_5(A)$	$\emptyset'' \leq_{\mathrm{T}} A \oplus K$
$[a,b]BC^n$	$U_4(A)$	$A \leq_{\mathrm{T}} K$	$U_5(A)$	$\emptyset'' \leq_{\mathrm{T}} A \oplus K$

Table 1: Characterization of the sets A such that, for the various inference types $\mathcal{I}$ listed in the far-left column, $\mathcal{I}[A] = \mathcal{I}$ (TRIV), $\mathcal{I}[A*] = \mathcal{I}$ ($*$-TRIV), $REC \in \mathcal{I}[A]$ (OMNI), $REC \in \mathcal{I}[A*]$ ($*$-OMNI), respectively.

7.2 Structure of the Degrees of Inferability

If B is high then $REC \in EX[B]$. Hence all the high degrees are of the same degree of inferability; thus we are only interested in the structure of the degrees of inferability of sets that are not high.

Theorem 7.13 ([KS96])

1. *If A, B are non-high r.e. sets then $A \leq_{\mathrm{i}} B$ iff $A \leq_T B$. (This also works if instead of EX inference we used BC inference, EX-team inference, or BC-team inference.)*

2. *If A, B are sets such that B is not high then $A \leq_{\mathrm{i}} B \Rightarrow A' \leq_T B'$. (Note that this holds for all sets, not just r.e. sets.)*

3. *Every non-omniscient EX-degree contains infinitely many Turing degrees.*

If $K' \leq_{\mathrm{T}} B \oplus K$ then $REC \in EX[B*]$. Hence all such Turing degrees are of the same $*$-degree of inferability; thus we are only interested in sets B for which $K' \not\leq_{\mathrm{T}} B \oplus K$.

Theorem 7.14 *Let A, B be sets such that $K' \not\leq_{\mathrm{T}} B \oplus K$. $A \leq_{\mathrm{i}}^{*} B$ iff $A \oplus K \leq_{\mathrm{T}} B \oplus K$. (This also works if instead of EX inference we used BC inference, EX-team inference, or BC-team inference.)*

Open Problem 7.15 Find a nice characterization of when $A \leq_{\mathrm{i}} B$.

7.3 Strength of Non-inclusion

How strong is the non-inclusion $BC \not\subseteq EX$? There are several ways this question can be phrased. One of them is to ask how powerful a set A would have to be such that $BC \subseteq EX[A]$. In this section we investigate several theorems in inductive inference in this light. Before we proceed we need the following definition.

Definition 7.16 Let PA denote the class of all Turing degrees containing a complete and consistent extension of Peano Arithmetic. See [Odi89, p. 510 ff.] for background information. It is known that PA coincides with the degrees of functions in $\{g\colon \mathsf{N} \to \{0,1\} \mid (\forall i)[g(i) \neq \varphi_i(i)]\}$ (see [Joc89, Proposition 2]). By the Low Basis Theorem [JS72] there exist degrees in PA that are low.

The following theorem measures how 'strong' some known theorems in inductive inference are.

Theorem 7.17 *[Kum94, KS96] Let $a \in \mathsf{N}$ and $n \geq 1$.*

1. *There is no oracle A such that $EX_{a+1} \subseteq EX_a[A]$.*
2. *$EX^{n+1} \subseteq EX^n[A]$ iff A is high.*
3. *$BC \subseteq EX^*[A]$ iff A is high.*

4. $[1, n+1]EX \subseteq [1, n]EX[A]$ *iff* A *is high.*

5. $[n+1, 2n+1]EX_0 \subseteq [n, 2n-1]EX_0[A]$ *iff* $K \leq_{\mathrm{T}} A$.

6. $[2, 4]EX_0 \subseteq [1, 2]EX_0[A]$ *iff* $K \leq_{\mathrm{T}} A$.

7. $[24, 49]EX_0 \subseteq [2, 4]EX_0[A]$ *iff the Turing degree of* A *is in* PA. *In particular there is a low oracle* A *such that* $[24, 49]EX_0 \subseteq [2, 4]EX_0[A]$.

Proof sketch:

1) The class in $EX_1 - EX_0[A]$ is

$$\{0^\omega\} \cup \{0^i 1^\omega : i \in \mathsf{N}\}.$$

2,3,4) One can construct a family $\mathcal{S} = \{\varphi_{g(i)}\}_{i \in \omega}$ ($g \in REC$) with the following properties:

(a) $1^i 0 \preceq \varphi_{g(i)}$.

(b) $\varphi_{g(i)}$ is defined for all x with at most one exception $a_i > i$.

(c) If $\varphi_{g(i)} \subseteq \varphi_j$, φ_j is total, and W_i is finite, then $j > |W_i|$.

The class $\mathcal{S} = \{f \in REC_{0,1} : (\exists i)[f \text{ extends } \varphi_{g(i)}]\}$ is obviously in EX^1. One can then show that if $\mathcal{S} \in EX[A]$ then that A is high.

5) Consider the class

$$\mathcal{S} = \{i0^\omega : i \notin K\} \cup \{i0^t j^\omega : i \in K_t - K_{t-1} \wedge j \leq n\}.$$

We show $\mathcal{S} \in [n+1, 2n+1]EX_0$. Upon seeing the first input i, output programs $p_0, \ldots, p_n$ such that program p_k will do the following: Keep outputting 0's until you spot that $i \in K_t$ (if this never happens then $i \notin K$ and the $n+1$ programs already defined suffice). If $i \in K_t$ then program p_k will be $i0^t k^\omega$. Note that one of these programs is correct. There are still n programs that have not been started yet. These all know that $i \in K$ and they know the correct value of j. They all produce $i0^t j^\omega$.

Since $\mathcal{S} \in [n+1, 2n+1]EX_0$ we have $\mathcal{S} \in [n, 2n-1]EX_0[A]$ via OIIMs $M_1^A, \ldots, M_{2n-1}^A$. We use this to obtain $K \leq_{\mathrm{T}} A$. Given i we determine if $i \in K$ (recursively in A) as follows. Find the least t such that on input $i0^t$ at least n of $M_1^A, \ldots, M_{2n-1}^A$ output an index. It is easy to see that $i \in K_t$ iff $i \in K$.

6) Let $a_0, a_1, \ldots,$ be a recursive 1-1 enumeration of K. Let

$$\begin{aligned}\mathcal{S} \;=\; & \{ij0^\omega : i,j \notin K\} \cup \\ & \{ij0^t1^\omega, ij0^\omega : (i = a_t \wedge j \notin K) \vee (j = a_t \wedge i \notin K)\} \cup \\ & \{ij0^t1^s0^\omega, ij0^t1^\omega, ij0^t0^s1^\omega, ij0^\omega : \{i,j\} = \{a_t, a_s\} \wedge t \leq s\}.\end{aligned}$$

It is not hard to show that $\mathcal{S} \in [2,4]EX_0$ and hence it is in $[1,2]EX_0[A]$. From this one can show that $K \leq_{\mathrm{T}} A$. The proof is unusual in that it is not a direct algorithm. It uses the following notions.

Definition 7.18 [BGGO93] Let X and Y be sets and $a, b \geq 1$. C_a^X is the function $C_a^X(x_1, \ldots, x_a) = X(x_1) \cdots X(x_a)$ $(\in \{0,1\}^a)$. A function f is *b-enumerable* if there is a Turing machine M that will, on input x, output (over time) at most b numbers such that one of them is $f(x)$. It is easy to see that C_a^X is 2^a-enumerable. A function f is *b-enumerable-in-Y* if there is an oracle Turing machine M^Y that will, on input x, output (over time) at most b numbers such that one of them is $f(x)$.

It is known [BGGO93] that if C_a^X is a-enumerable then X is recursive. This easily relativizes to show that if C_a^X is a-enumerable-in-Y then $X \leq_{\mathrm{T}} Y$. One can show that if $\mathcal{S} \in [1,2]EX_0$ then C_2^K is 2-enumerable-in-A, hence $K \leq_{\mathrm{T}} A$.

7) The proof that $[24,49]EX_0 \subseteq [2,4]EX_0[A] \Rightarrow deg(A) \in PA$ is a clever and complicated modification of the proof that $[24,49]EX_0 \not\subseteq [2,4]EX_0$ from [DKV95]. It should be noted that the original proof used the Recursion Theorem, while this proof does not.

The proof that $deg(A) \in PA \Rightarrow [24,49]EX_0 \subseteq [2,4]EX_0[A]$ uses the fact that every partial recursive 0-1 valued function has an extension recursive in A. ∎

Theorem 7.17 can be interpreted as follows.

1. The result $EX_{a+1} \not\subseteq EX_a$ is very strong in that it is impossible to break it with an oracle.

2. The results $EX^{n+1} \not\subseteq EX^n$, $BC \not\subseteq EX$, and $[1,n+1]EX \not\subseteq [1,n]EX$ are all very strong since in order to break them you need to supply an oracle that already allows REC to be inferred.

3. The result $[2,4]EX_0 \not\subseteq [1,2]EX_0$ is very strong since in order to break it you need an oracle as hard as K.

4. The result $[24,49]EX_0 \not\subseteq [2,4]EX_0$ is not very strong since a low oracle suffices to break it.

Open Problem 7.19 Classify non-inclusion results in inductive inference via the set of oracles that make them fail. Try to find a coherent explanation for the classes obtained.

8 Query Inference: Asking Both

What happens if we allow the learner to make queries about the function *and* queries to an oracle?

Definition 8.1 [Ste95] Let $\mathcal{S} \subseteq REC$. Let L be a query language. $\mathcal{S} \in QEX[L; A]$ if there is an oracle QIM that infers any $f \in \mathcal{S}$ using queries in L about f and queries to A.

Theorem 8.2 *[Ste95] If* $EX[A] \not\subseteq EX[B]$ *then*

1. $QEX[+, <; A] \not\subseteq QEX[+, <; B]$.
2. $QEX[<; A] \not\subseteq QEX[<; B]$.
3. $QEX[Succ; A] \not\subseteq QEX[Succ; B]$.

Proof sketch: This proof uses k-good sets [GPS92]. A set is k-good if, informally, the elements in it are very far apart (how far apart is parameterized by k). If a set is k-good then queries about it in $[+, <]$ can be answered by just knowing k. The sets that are constructed for the non-inclusions are k-good sets so that the ability to ask queries in $[+, <]$ does not help. ∎

Theorem 8.3 $REC \in QEX[Succ; A] \cup QEX[<; A] \cup QEX[+, <; A]$ *iff* A *is high.*

By the above theorem the combination of queries and oracles is not that powerful for inferring REC: if you can infer REC then the oracle must be doing all the work. The next theorem shows that the combination of queries and oracles can be powerful for inferring some classes of functions.

Theorem 8.4 *There is a set A such that*

1. $EX = EX[A]$,
2. $QEX[+,<] \subset QEX[+,<;A]$,
3. $QEX[<] \subset QEX[<;A]$, *and*
4. $QEX[Succ] \subset QEX[Succ;A]$.

Open Problem 8.5 Prove theorems about $QEX[[Succ,<]^2;A]$. This is particularly interesting since, by Lemma 6.12 and reductions from $[Succ,<]^2$, we could obtain results about many other inference types.

9 Conclusions

Inductive inference has revealed many intuitions concerning machine learning by example. In this survey we have formalized the intuition that asking questions helps one to learn. In the survey of [GS95] the theme of formalizing aspects of human learning is pursued in more depth.

Inductive inference has also served as a rich application area for recursion theory and many other subareas of mathematical logic. We surveyed the area of learning via queries to highlight since this shows off many of these areas. We list the techniques used to prove theorems in this survey. The references given refer to a place where the technique is used; however, in most cases, the technique is used in several places.

1. Diagonalization (Theorem 4.1).
2. Non-constructive proofs (Theorem 4.1.3 and 4.1.4).
3. The Recursion Theorem (Theorem 4.2).
4. Complex simulation (Theorem 4.3).

5. Probability (Theorem 4.5).
6. Ordinals (Theorem 4.9 and also [FS93]).
7. Hilbert's 10th problem (Theorem 6.1).
8. Büchi automata (Theorem 6.8).
9. The Operator Recursion Theorem (Theorem 6.16).
10. Decidable theories (Theorem 6.14).
11. More recursion theory than you want to know (the table in Section 7.1).
12. Bounded queries (Theorem 7.17.6).

We again state what we consider to be the most important open problem in the field; we also repeat our previous commentary on this problem. Our conjecture is similar to the fact that the arithmetic hierarchy is proper.

Open Problem 9.1 Do more alternations of quantifiers help? Conjectures:

1. $Q_1EX[Succ, <] \subset Q_2EX[Succ, <] \subset Q_3EX[Succ, <] \cdots$.
2. $Q_1EX[+, <] \subset Q_2EX[+, <] \subset Q_3EX[+, <] \cdots$.

It is known that the $Q_iEX[Succ, <]$[2] hierarchy collapses since every query is equivalent to a Büchi automaton, which is in turn equivalent to a (second order) formula with four alternations of quantifiers. It is also known that for all $i \geq 1$ $Q_iEX[+, \times] = Q_{i+1}EX[+, \times]$ since $REC \in Q_1EX[+, \times]$. Therefore, if the conjecture is true, the proof will have to use particular properties of $[Succ, <]$ and $[+, <]$.

10 Acknowledgments

We would like to thank Kalvis Apsitis, Mark Changizi, Andrew Lee, and Georgia Martin for proofreading and commentary.

References

[AB91] Lenny Adleman and Manuel Blum. Inductive inference and unsolvability. *Journal of Symbolic Logic*, 56(3):891–900, September 1991.

[AFS] Kalvis Apsitis, Rūsiņš Freivalds, and Carl H. Smith. On duality in learning and the selection of learning team. *Information and Computation.* To appear. Earlier version appeared in STOC94 under the title *Choosing a learning team: a topological approach.*

[AFS95] Kalvis Apsitis, Rūsiņš Freivalds, and Carl H. Smith. On the inductive inference of real valued functions. In *Proc. 8th Annu. Conf. on Comput. Learning Theory*, pages 170–177. ACM Press, New York, NY, 1995.

[Amb96] Andris Ambainis. Probabilistic PFIN-type learning: General properties. In *Proceedings of the Ninth Annu. Conference on Computational Learning Theory, Santa Cruz, California.* ACM Press, 1996.

[AS83] Dana Angluin and Carl H. Smith. A survey of inductive inference: Theory and methods. *ACM Comput. Surv.*, 15(3):237–269, September 1983.

[Bar74] Janis Barzdins. Two theorems on the limiting synthesis of functions. *Theory of Algorithms and Programs, Latvian State University, Riga*, pages 82–88, 1974. In Russian.

[Bar80] Janis Barzdins. Barzdins' conjecture, 1980. Private Communication.

[BB75] Lenore Blum and Manuel Blum. Towards a mathematical theory of inductive inference. *Information and Control*, 28:125–155, 1975.

[BC94] Ganesh Baliga and John Case. Learning with higher order additional information. In *Proc. 5th Int. Workshop on Algorithmic Learning Theory*, pages 64–75. Springer-Verlag, 1994.

[BGGO93] Richard Beigel, William Gasarch, John T. Gill, and James C. Owings. Terse, superterse, and verbose sets. *Information and Computation*, 103:68–85, 1993.

[Büc60] J. Richard Büchi. Weak second order arithmetic and finite automata. *Zeitsch. f. math. Logik und Grundlagen d. Math.*, 6:66–92, 1960.

[Büc62] J. Richard Büchi. On a decision method in restricted second-order arithmetic. In *Proc. of the International Congress on logic, Math, and Philosophy of Science (1960)*. Stanford University Press, 1962.

[Cas74] John Case. Periodicity in generations of automata. *Mathematical Systems Theory*, 8:15–32, 1974.

[Che81] Keh Jiann Chen. *Tradeoffs in Machine Inductive Inference*. PhD thesis, Computer Science Department, SUNY at Buffalo, 1981.

[Che82] Keh Jiann Chen. Tradeoffs in the inductive inference of nearly minimal size programs. *Inform. Control*, 52:68–86, 1982.

[Cho74] Yaacov Choueka. Theories of automata on ω-tapes: A simplified approach. *Journal of Computer and System Sciences*, 8:117–142, 1974.

[CJM94] John Case, Sanjay Jain, and S. Ngo Manguelle. Refinements of inductive inference by popperian and reliable machines. *Kybernetika*, 30–1:23–52, 1994.

[CL82] John Case and Christopher Lynes. Machine inductive inference and language identification. In M. Nielsen and E. M. Schmidt, editors, *Proceedings of the 9th International Colloquium on Automata, Languages and Programming*, pages 107–115. Springer-Verlag, 1982. Lecture Notes in Computer Science 140.

[CM79] John Case and S. Ngo Manguelle. Refinements of inductive inference by Popperian machines. Technical Report 152, SUNY/Buffalo, 1979.

[CM95] Douglas A. Cenzer and William R. Moser. Inductive inference of functions on the rationals. In *Proc. 8th Annu. Conf. on Comput. Learning Theory*, pages 178–181. ACM Press, New York, NY, 1995.

[CS83] John Case and Carl H. Smith. Comparison of identification criteria for machine inductive inference. *Theoretical Computer Science*, 25:193–220, 1983.

[DKV95] Robert Daley, Bala Kalyanasundaram, and Mahendra Velauthapillai. Breaking the probability 1/2 barrier in FIN-type learning. *Journal of Computer and System Sciences*, 50:574–599, 1995.

[DPR61] Martin Davis, Hillary Putnam, and Julia Robinson. The decision problem for exponential diophantine equations. *Annals of Mathematics*, 74:425–436, 1961.

[DPVW91] Robert Daley, Lenny Pitt, Mahendra Velauthapillia, and Todd Will. Relations between probabilistic and team one-shot learners. In *Proc. 4th Annu. Workshop on Comput. Learning Theory*, pages 228–239, San Mateo, CA, 1991. Morgan Kaufmann.

[DS86] Robert Daley and Carl H. Smith. On the complexity of inductive inference. *Information and Control*, 69:12–40, 1986.

[FJ89] Mark Fulk and Sanjay Jain. Learning in the presence of inaccurate information. In *Proceedings of the Second Annu. Workshop on Computational Learning Theory, Santa Cruz, California*, pages 175–188. Morgan Kaufmann, Inc., August 1989.

[FJ95] Rūsiņš Freivalds and Sanjay Jain. Kolmogorov numberings and minimal identification. In *Proceedings of the Second European Conference on Computational Learning Theory, Barcelona, Spain*, March 1995.

[FJG+94] Lance Fortnow, Sanjay Jain, William Gasarch, Efim Kinber, Martin Kummer, Stuart Kurtz, Mark Pleszkoch, Theodore Slaman, Frank Stephan, and Robert Solovay. Extremes in the de-

grees of inferability. *Annals of pure and applied logic*, 66:21–276, 1994.

[FKS94] Rūsiņš Freivalds, Marek Karpinski, and Carl H. Smith. Co-learning of total recursive functions. In *Proc. 7th Annu. ACM Workshop on Comput. Learning Theory*, pages 190–197. ACM Press, New York, NY, 1994.

[FKS95] Rūsiņš Freivalds, Efim Kinber, and Carl H. Smith. On the impact of forgetting on learning machines. *Journal of the ACM*, 42(6):1146–1168, 1995.

[FKS96] Rūsiņš Freivalds, Efim Kinber, and Carl H. Smith. On the intrinsic complexity of learning. *Information and Computation*, 123(2):64–71, 1996.

[FKW95] Rūsiņš Freivalds, Efim Kinber, and Rolf Wiehagen. How inductive inference strategies discover their errors. *Inform. Control*, 118:208–226, 1995.

[Fre79] Rūsiņš Freivalds. Finite identification of general recursive functions by probabilistic strategies. In L. Budach, editor, *Proceedings of the Conference on Algebraic, Arithmetic, and Categorial Methods in Computation Theory*, pages 138–145, Berlin, 1979.

[FS93] Rūsiņš Freivalds and Carl H. Smith. On the role of procrastination for machine learning. *Inform. Comput.*, 107(2):237–271, 1993.

[FS97] Rūsiņš Freivalds and Carl H. Smith. On the duality between mechanistic learners and what it is they learn. *Information and Computation*, 1997. To appear. An earlier version appeared in the Proc. 4th Int. Workshop on Algorithmic Learning Theory, 1993, 137–149.

[FSV89] Rūsiņš Freivalds, Carl H. Smith, and Mahendra Velauthapillai. Trade-off among parameters affecting inductive inference. *Information and Computation*, 82(3):323–349, September 1989.

[Ful90a] Mark Fulk. Prudence and other conditions on formal language learning. *Information and Computation*, 85:1–11, 1990.

[Ful90b] Mark Fulk. Robust separations in inductive inference. In *Proc. of the 31st IEEE Sym. on Found. of Comp. Sci.*, pages 405–410. IEEE Computer Society Press, Los Alamitos, CA, 1990.

[GH95] William Gasarch and Geoffrey R. Hird. Reductions for learning via queries. In *Proc. 8th Annu. Conf. on Comput. Learning Theory*, pages 152–161. ACM Press, New York, NY, 1995.

[Gil77] John Gill. Computational complexity of probabilistic Turing machines. *SIAM Journal on Computing*, 6:675–695, 1977.

[GKP+95] William Gasarch, Efim Kinber, Mark Pleszkoch, Carl H. Smith, and Thomas Zeugmann. Learning via queries, teams, and anomalies. *Fundamenta Informaticae*, 23:67–89, 1995. Shorter version in Third Annu. Conference on Computational Learning Theory, 1990, pages 327-337, published by Morgan Kaufman.

[Gol67] E. Mark Gold. Language identification in the limit. *Information and Control*, 10(10):447–474, 1967.

[GP89] William Gasarch and Mark Pleszkoch. Learning via queries to an oracle. In *Second Annu. Conference on Computational Learning Theory*, pages 214–229. Morgan Kaufman, 1989.

[GPS92] William Gasarch, Mark Pleszkoch, and Robert Solovay. Learning via queries to $[+, <]$. *Journal of Symbolic Logic*, 57(1):53–81, March 1992.

[GPSVar] William Gasarch, Mark Pleszkoch, Frank Stephan, and Mahendra Velauthapillai. Classification using information. *Annals of Math and AI*, to appear. Also appeared in Proc. 5th Int. Workshop on Algorithmic Learning Theory, 1994, 290–300.

[GS92] William Gasarch and Carl H. Smith. Learning via queries. *Journal of the ACM*, 39(3):649–675, July 1992. A shorter version is in 29th FOCS conference, 1988, pp. 130-137.

[GS95] William Gasarch and Carl H. Smith. Recursion theoretic models of learning: some results and intuitions. *Annals of Mathematics and Artificial Intelligence*, 15:151–166, 1995.

[GSSV92] William Gasarch, Ramesh Sitaraman, Carl H. Smith, and Mahendra Velauthapillai. Learning programs with an easy to calculate set of errors. *Fundamentica Informaticae*, pages 355–370, 1992.

[GV93] William Gasarch and Mahendra Velauthapillai. Asking questions versus verifiability. In *International Workshop on Analogical and Inductive Inference*, volume 642 of *Lecture Notes in Computer Science*, pages 197–213. Springer-Verlag, 1993.

[Jai94] Sanjay Jain. Program synthesis in the presence of infinite number of inaccuracies. In *Algorithmic Learning Theory, Proceedings of the 4th International Workshop on Analogical and Inductive Inference, AII'94 and 5th International Workshop on Algorithmic Learning Theory, ALT'94, Reinhardsbrunn Castle, Germany*, pages 333–348. Springer Verlag, October 1994. Lecture Note in AI 872.

[Jai95] Sanjay Jain. On a question about learning nearly minimal programs. *Inform. Proc. Lett.*, 53, 1995.

[Joc80] Carl G. Jockusch. Degrees of generic sets. In F.R. Drake and S.S. Wainer, editors, *Recursion Theory: its generalizations and applications*, pages 110–139. Cambridge University Press, 1980.

[Joc89] Carl G. Jockusch. Degrees of functions with no fixed points. In *Logic, Methodology, and Philosophy of Science VIII*, pages 191–201, 1989.

[JS72] Carl G. Jockusch and Robert I. Soare. Π^0_1 classes and degrees of theories. *Transactions of the AMS*, 173:33–56, 1972.

[JS90] Sanjay Jain and Arun Sharma. Language learning by a team. In M. S. Paterson, editor, *Proceedings of the 17th International Colloquium on Automata, Languages and Programming*, pages

153–166. Springer-Verlag, July 1990. Lecture Notes in Computer Science, 443.

[JS93a] Sanjay Jain and Arun Sharma. Learning with the knowledge of an upper bound on program size. *Inform. Comput.*, 102–1:118–166, 1993. Presented at the Workshop on Computational Learning Theory and Natural Learning Systems.

[JS93b] Sanjay Jain and Arun Sharma. On the non-existence of maximal inference degrees for language identification. *Information Processing Letters*, 47(2):81–88, 1993.

[JS93c] Sanjay Jain and Arun Sharma. Probability is more powerful than team for language identification. In *Proceedings of the Sixth Annu. Conference on Computational Learning Theory, Santa Cruz, California*, pages 192–198. ACM Press, July 1993.

[JS94a] Sanjay Jain and Arun Sharma. On the intrinsic complexity of language identification. In *Proc. 7th Annu. ACM Workshop on Comput. Learning Theory*, pages 278–286. ACM Press, New York, NY, 1994.

[JS94b] Sanjay Jain and Arun Sharma. Program size restrictions in computational learning. *Theoret. Comput. Sci. A*, 127:351–386, 1994.

[JS95a] Sanjay Jain and Arun Sharma. On aggregating teams of learning machines. *Theoretical Computer Science*, 137(1):85–108, January 1995.

[JS95b] Sanjay Jain and Arun Sharma. The structure of intrinsic complexity of learning. In *Proceedings of the Second European Conference on Computational Learning Theory, Barcelona, Spain*, March 1995.

[JS96] Sanjay Jain and Arun Sharma. Generalization and specialization strategies for identifying r .e. languages. *Annals of Mathematics and Artificial Intelligence*, 1996. To appear.

[JSV95] Sanjay Jain, Arun Sharma, and Mahendran Velauthapillai. Finite identification of functions by teams with success ratio $\frac{1}{2}$ and above. *Information and Computation*, 121–2:201–213, 1995.

[KS95] Efim Kinber and Frank Stephan. Language learning from texts: Mind changes, limited memory and monotonicity. *Information and Computation*, 123(2):224–241, 1995.

[KS96] Martin Kummer and Frank Stephan. On the structure of the degrees of inferability. *Journal of Computer and System Sciences*, 52(2):214–238, 1996. Earlier version in Sixth Annu. Conference on Computational Learning Theory, 1993.

[Kum94] Martin Kummer. The strength of noninclusions for teams of finite learners. In *Proc. 7th Annu. ACM Workshop on Comput. Learning Theory*, pages 268–277. ACM Press, New York, NY, 1994.

[Mat70] Yuri Matijasevic. Enumerable sets are diophantine (Russian). *Doklady Academy Nauk, SSSR*, 191:279–282, 1970. Translation in Soviet Math Doklady, Vol 11, 1970.

[Mat93] Yuri Matijasevic. *Hilbert's Tenth Problem.* MIT press, Cambridge, 1993.

[Odi89] Piergiorgio Odifreddi. *Classical Recursion Theory (Volume I).* North-Holland, Amsterdam, 1989.

[OSW82] Daniel Osherson, Michael Stob, and Scott Weinstein. Learning strategies. *Information and Control*, 53:32–51, 1982.

[OSW86a] Daniel Osherson, Michael Stob, and Scott Weinstein. Aggregating inductive expertise. *Information and Computation*, 70:69–95, 1986.

[OSW86b] Daniel Osherson, Michael Stob, and Scott Weinstein. *Systems that Learn: An Introduction to Learning Theory for Cognitive and Computer Scientists.* MIT Press, 1986.

[OW82] Daniel Osherson and Scott Weinstein. Criteria of language learning. *Information and Control*, 52:123–138, 1982.

[Pit89] Lenny Pitt. Probabilistic inductive inference. *Journal of the ACM*, 36(2):383–433, 1989. Earlier version in 1984 STOC conference. Part of his 1985 Yale Ph.D Thesis.

[Pod74] Karlis Podnieks. Comparing various concepts of function prediction, Part 1. *Theory of Algorithms and Programs, Latvian State University, Riga*, pages 68–81, 1974. In Russian.

[Pod75] Karlis Podnieks. Probabilistic synthesis of enumerated classes of functions. *Soviet Math. Doklady*, 16:1042–1045, 1975.

[Pod77] Karlis Podnieks. Probabilistic program synthesis. *Theory of Algorithms and Programs, Latvian State University, Riga*, 16:57–88, 1977.

[PS88] Lenny Pitt and Carl H. Smith. Probability and plurality for aggregations of learning machines. *Information and Computation*, 77:77–92, 1988.

[Put75] Hillary Putnam. Probability and confirmation. In *Mathematics, Matter and Method*, volume 1. Cambridge University Press, 1975. Based on Radio Lectures given in 1967.

[Rab77] Michael O. Rabin. Decidable theories. In Jon Barwise, editor, *Handbook of Mathematical Logic*. North Holland, 1977.

[Roy86] James Royer. Inductive inference of approximations. *Inform. Control*, 70:156–178, 1986.

[Saf88] Shumuel Safra. On the complexity of ω-automata. In *Proc. of the 29th IEEE Sym. on Found. of Comp. Sci.*, pages 319–329, 1988.

[Smi82] Carl H. Smith. The power of pluralism for automatic program synthesis. *Journal of the ACM*, 29(4):1144–1165, October 1982. Was also in FOCS 1981.

[Smi94a] Carl H. Smith. *A Recursive Introduction to the Theory of Computation.* Springer-Verlag, 1994.

[Smi94b] Carl H. Smith. Three decades of team learning. In *Proceedings of AII/ALT'94, Lecture Notes in Artificial Intelligence*, volume 872, pages 211–228. Springer Verlag, 1994.

[Soa87] Robert I. Soare. *Recursively Enumerable Sets and Degrees.* Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1987.

[SS91] Theodore Slaman and Robert Solovay. When oracles do not help. In *Fourth Annu. Conference on Computational Learning Theory*, pages 379–383. Morgan Kaufman, 1991.

[Ste95] Frank Stephan. Learning via queries and oracles. In *Proc. 8th Annu. Conf. on Comput. Learning Theory*, pages 162–169. ACM Press, New York, NY, 1995. Submitted to Annals of Pure and Applied Logic.

[SV90] Carl H. Smith and Mahendra Velauthapillai. On the inference of approximate programs. *Theoretical Computer Science*, 77, 1990.

[Vel89] Mahendra Velauthapillai. Inductive inference with bounded number of mind changes. In *Proc. 2nd Annu. Workshop on Comput. Learning Theory*, pages 200–213, San Mateo, CA, 1989. Morgan Kaufmann.

[WFK84] Rolf Wiehagen, Rūsiņš Freivalds, and Efim Kinber. On the power of probabilistic strategies in inductive inference. *Theoret. Comput. Sci.*, 28:111–133, 1984.

[Wie78a] Rolf Wiehagen. Characterization problems in the theory of inductive inference. In *Proc. of 5th Colloquium on Automata, Languages, and Programming*, pages 494–508. Springer-Verlag, 1978.

[Wie78b] Rolf Wiehagen. *Zur Theorie der Algorithmischen Erkennung.* PhD thesis, Humboldt-Universitat, Berlin, 1978.

[WS95] Rolf Wiehagen and Carl H. Smith. Generalization versus classification. *Journal of Experimental and Theoretical Artificial Intelligence*, 7, 1995. Also in fifth COLT conference.

[ZL95] Thomas Zeugmann and Stephan Lange. A guided tour across the boundaries of learning recursive languages. In *Algorithmic Learning for Knowledge-Based Systems*, volume Lecture Notes in Artificial Intelligence 961, pages 193–262. Springer–Verlag, 1995.