Learning via Queries in $[+,<]$

by

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

Mark G. Pleszkoch[2]
Department of Computer Science
University of Maryland
College Park, MD 20742

Robert Solovay[3]
Department of Mathematics
University of California
Berkeley, CA 94720

ABSTRACT

We prove that the set of all recursive functions cannot be inferred using first-order queries in the query language containing extra symbols $[+,<]$. The proof of this theorem involves a new decidability result about Presburger arithmetic which is of independent interest. Using our machinery, we show that the set of all primitive recursive functions cannot be inferred with a bounded number of mind changes, again using queries in $[+,<]$. Additionally, we resolve an open question in [7] about passive versus active learning.

1) Introduction

This paper presents new results in the area of query inductive inference (introduced in [7]); in addition, there are results of interest in mathematical logic.

Inductive inference is the study of inductive machine learning in a theoretical framework. In query inductive inference, we study the ability of a Query Inference Machine

[1] Supported, in part, by NSF grants CCR 88-03641 and 90-20079.
[2] Also with IBM Corporation, Application Solutions Division, Gaithersburg, MD 20879.
[3] Supported, in part, by NSF grant DMS 87-01828.

(QIM) to learn a recursive function f that is chosen from a given concept class (i.e. set) of recursive functions. A QIM is a Turing machine with the ability to ask questions about f in a particular query language (and magically get the answers). This differs from passive inductive inference [1], where the inductive inference machine (IIM) is a Turing machine that is presented with the function values $f(0)$, $f(1)$, $f(2), \ldots$, one after another, and has no other way of obtaining additional information about the function. Both query and passive inductive inference deal with inference in the limit; that is, the inference machines execute forever, possibly always asking new questions or receiving new function values. From time to time, the inference machine may make a conjecture (guess) about the function f, in the form of a program which is supposed to compute f. The guess can be null, denoted $\perp$. If the inference machine guesses the same program from some point on, and that program is correct, then we say that the inference machine has *inferred* the function f. We denote by EX the set of concept classes of recursive functions which can be inferred in the limit by an IIM.

In what follows, we assume familiarity with [7] and use its notation. A key observation of Gasarch and Smith [7] was that, for query inductive inference, the query language used plays an important role in determining which concept classes of recursive functions can be inferred. For the most part, the query languages studied consist of first-order sentences containing the standard logical symbols ($\wedge$,$\vee$,$\neg$,$\exists$,$\forall$,$=$), constant symbols for each natural number, a function symbol $\mathcal{F}$ representing the function to be inferred, and extra symbols such as $[+, \times]$ or $[+, <]$. If recursive sets are being inferred instead of functions, then a predicate symbol $\mathcal{A}$ is used in place of the function symbol $\mathcal{F}$.

Let L be the extra symbols in a language, and let $a, i \in \mathbb{N}$ ($\mathbb{N}$ is the set of natural numbers). We denote by QEX$[L]$ (resp., Q$_i$EX$[L]$) the set of concept classes of recursive functions which can be inferred in the limit by QIMs using the query language L (resp., the subset of queries using L that have at most i alternations of quantifiers and begin with a $\exists$). That is, if C is a set of recursive functions, then $C \in$ QEX$[L]$ if and only if there exists a QIM using the query language with extra symbols L that can infer every function $f \in C$. We denote by QEX$_a[L]$ the set of concept classes of recursive functions which can be inferred in the limit by QIMs using the query language L, which only change their mind at most a times. A machine that is trying to $\mathrm{QEX}_a[L]$-infer a function may repeatedly output $\perp$ up until, but not after, it has made its first non-null guess. The change from $\perp$ to a non-null guess is not counted as a mind change. The class $\mathrm{Q}_i\mathrm{EX}_a[L]$ can easily be defined by combining these two bounds.

Let REC ($RECSET$) denote the set of all recursive functions (sets). It is known that $REC \in \mathrm{QEX}[+,\times]$ (in fact $REC \in \mathrm{Q}_1\mathrm{EX}[+,\times]$ and $REC \in \mathrm{Q}_2\mathrm{EX}_0[+,\times]$ by Theorems 9, 10 of [7]), but that $REC \notin \mathrm{QEX}[S,<]$ (Theorem 23 of [7]), where S denotes the successor function. These facts lead to the conjecture that $REC \notin \mathrm{QEX}[L]$ for any decidable query language L, using an appropriate notion of query language decidability. This conjecture is supported by Theorem 13 of [7] which shows that, for any query language whose existential theory is decidable when the function symbol $\mathcal{F}$ is removed, $REC \notin \mathrm{Q}_1\mathrm{EX}[L]$.

Thus, we would suspect that $REC \notin \mathrm{QEX}[+,<]$, based on the fact that Presburger arithmetic is decidable (see [3] for a proof of this). However, since the queries in $[+,<]$ are not strictly a part of Presburger arithmetic (as they contain the additional function symbol $\mathcal{F}$), it turns out that we require a new and stronger decidability result to show that $REC \notin \mathrm{QEX}[+,<]$. This new decidability result plays the same role in the proof of $REC \notin \mathrm{QEX}[+,<]$ as ω-automata did in the proof of $REC \notin QEX[S,<]$: To diagonalize against the QIM (Query Inference Machine), we need to be able to answer the queries in such a way that we always have an uncountable number of total functions consistent with the query answers to date. That we can do this for $[+,<]$ is somewhat surprising in light of the following result from [4].

Proposition 1. It is undecidable to determine, for a general query q in $[+,<]$, whether or not there are an uncountable number of total functions which make the query true. In fact, it is undecidable to determine if there exists any total function which makes the query true.

This proposition still holds if we restrict ourselves to 0-1 valued functions, i.e., sets. Recall that these problems were decidable for $[S,<]$, using ω-automata. Fortunately, the essential role of decidability in diagonalizing against a QIM is contained in the following corollary of the main decidability result which we prove below.

- (Section 2) It is decidable, for a general query q in $[+,<]$, to select one of q or $\neg q$ such that there are an uncountable number of functions which make the selected query true.

We use the machinery used to prove this theorem, and the theorem itself, to show

- (Section 3) $REC \notin \mathrm{QEX}[+,<]$, and
- (Section 4) for all c, $PRIMREC \notin \mathrm{QEX}_c[+,<]$, where $PRIMREC$ is the set of all primitive recursive functions.

Readers who are only interested in the mathematical logic results need only read Section 2. Readers who are only interested in inductive inference need only know the statements, but not the proofs, of the results in Section 2.

A language is *reasonable* if all the extra symbols denote functions and predicates that are computable. In [7] it was shown that for any reasonable language L, $\mathrm{Q_1EX_0}[L] \subseteq \mathrm{EX}$. It was an open question to determine for which L the inclusion was proper. Using ω-automata, one can show that, for all c, $PRIMREC \notin \mathrm{QEX}_c[S,<]$ [7]. As noted above, the machinery of this paper can be used to show that, for all c, $PRIMREC \notin \mathrm{QEX}_c[+,<]$; since $PRIMREC \in \mathrm{EX}$ [8], these results demonstrate that $\mathrm{Q_1EX_0}[S,<] \subset \mathrm{EX}$ and $\mathrm{Q_1EX_0}[+,<] \subset \mathrm{EX}$. It is doubtful that similar machinery can be developed for $[+,\times]$ since $PRIMREC \in \mathrm{Q_1EX_0}[+,\times]$.

Using an entirely different approach, we resolve the question of how EX and $\mathrm{Q_1EX_0}[L]$ compare by showing:

- (Section 5) For any reasonable language L, and any $c \in \mathbb{N}$, $\mathrm{EX}_{c+1} - \mathrm{Q_1EX}_c[L] \neq \emptyset$, hence $\mathrm{Q_1EX_0}[L] \subset EX$.

The interested reader may consult [1,2,5,10] for further background on recursion-theoretic inductive inference.

2) Decidability Results

In this section, we present the decidability results, due to Solovay, that are used to prove $REC \notin QEX[+,<]$.

We make two modifications of the standard query inference framework. First, we extend the domain from the natural numbers to the integers. This does not affect the utility of the final decidability results, since the *less than* predicate allows us to specifically restrict the range of any variable to the natural numbers. Second, we use a set symbol $\mathcal{A}$ instead of a function symbol $\mathcal{F}$ in the queries. This actually increases the power of the inference result, as we prove $RECSET \notin QEX[+,<]$, instead of just $REC \notin QEX[+,<]$.

The essential idea in the proofs of the decidability results is that there is an uncountable class of sets (which we call k-good sets) for which it is decidable to determine if they satisfy a given query. The decision procedure itself uses a form of quantifier elimination to put the queries into a simple form. In our quantifier elimination, the original formula, $\psi_1(\mathcal{A})$ is guaranteed to be equivalent to the quantifier free version $\psi_2(\mathcal{A})$ only if a certain kind of set (a k-good set, where k is determined during quantifier elimination) is the interpretation of $\mathcal{A}$. We define the basic query language, and its interpretation in a set of models.

Definition: For an infinite set $S \subseteq \mathbb{N}$, the model $\mathbf{Z}_S$ (of the language L) has universe $\mathbf{Z}$ (the integers), where the non-logical symbols of L are given and interpreted as follows:

Symbol	Type	Interpretation
0	const	0
1	const	1
$+$	func 2	$\lambda x, y[x + y]$
$-$	func 1	$\lambda x[-x]$
$<$	pred 2	$\{\langle x, y\rangle : x < y\}$
$\mathcal{A}$	pred 1	$\{x : x \in S\}$

An L-formula is a first order formula (possibly containing free variables) constructed using the symbols of L and the logical symbols $\wedge$, $\vee$, $\neg$, $\exists$, and $\forall$. An L-sentence is an L-formula with no free variables. For example, $\exists x \mathcal{A}(x)$ is an L-sentence, and $\exists x \mathcal{A}(x) \wedge (x < y)$ is an L-formula, but not an L-sentence. Where convenient, we will occasionally use the *less than or equal to* predicate. In this event, we interpret $t_1 \leq t_2$ as an abbreviation for $t_1 < t_2 + 1$.

In order to perform quantifier elimination, we need to introduce extra operations to replace the expressive power that the quantifiers add.

Definition: The language L^{e} is an extension of the language L, where the new non-logical symbols are interpreted in $\mathbf{Z}_S$ as follows:

Symbol	Type	Interpretation
$\lfloor\cdot\rfloor$	func 1	$\lambda x[\max(\{0\} \cup \{y \in S : y \leq x\})]$
$\lceil\cdot\rceil$	func 1	$\lambda x[\min\{y \in S : y \geq x\}]$
D_2	pred 1	$\{x : (\exists y)\, 2y = x\}$
D_3	pred 1	$\{x : (\exists y)\, 3y = x\}$
D_4	pred 1	$\{x : (\exists y)\, 4y = x\}$
$\vdots$	$\vdots$	$\vdots$

We need one more langauge that will serve as an intermediary.

Definition: The language L_1 has logical symbols $0, 1, +, -$, and $<$; and also $D_2, D_3, D_4, \ldots$. They are intepreted as indicated in the above tables.

We now define the class of k-good sets, for which we actually decide queries in L and L^{e}.

Definition: For $k \geq 1$, a k-good set is an infinite set $S = \{s_0 < s_1 < s_2 < s_3 < \cdots\}$ satisfying:

1) $(\forall i \geq 0)$ $k!$ divides s_i,
2) $k < s_0$,
3) $(\forall i \geq 0)$ $k \cdot s_i < s_{i+1}$.

As usual, $k!$ denotes the product of the positive integers $\leq k$. Notice that there are an uncountable number of k-good sets for every k. Notice further that if $k' > k$, then every k'-good set is also a k-good set.

We give an intuition, and foreshadow, why k-good sets are useful for the decidability result. Assume that $z_1 > z_2 > \cdots > z_n$ are elements of a k-good set, where k is large. The number z_1 is quite large compared to the other elements. Therefore, statements like $\sum_i a_i z_i < t$, $\sum_i a_i z_i = t$, and $\sum_i a_i z_i > t$ may be determined purely by the relation of z_1 to t. In this manner, complicated statements reduce to easy ones.

The main decidability result is proved in two pieces, contained in the first two theorems of this section. The key part of the procedure is contained in the next theorem, which is the quantifier elimination step.

Theorem 2. There is an effective procedure that, given an L^{e}-formula q, produces a $k \geq 1$ and a quantifier-free L^{e}-formula q' such that for all k-good sets S and all variable assignments $\vec{v}$, we have:

$$\mathbf{Z}_S \models q[\vec{v}\,] \iff \mathbf{Z}_S \models q'[\vec{v}\,].$$

Theorem 2 is a direct consequence of Lemma 3.

Lemma 3. There is an effective procedure that given an L^{e}-formula of the form $\exists x\theta$, where θ is quantifier-free, produces a $k \geq 1$ and a quantifier-free L^{e} formula θ' such that for all k-good sets S and all variable assignments $\vec{v}$, we have:

$$\mathbf{Z}_S \models (\exists x)\theta[\vec{v}\,] \iff \mathbf{Z}_S \models \theta'[\vec{v}\,].$$

Proof:

To prove Lemma 3 we begin by describing a certain game G, played between two players I and II. A simple argument will establish that a recursive winning strategy for I in the game G yields a procedure $\mathcal{L}$ meeting the demands of Lemma 3. The bulk of our argument will consist in describing such a recursive winning strategy for player I.

The game G starts with player II choosing a formula of L^{e} of the form $\exists x\theta$ with θ quantifier-free. At any turn of the game, there will be a "current main formula". At the start of the game, it is the formula chosen by II. Intuitively, player I is trying to prove that there exists a k and a quantifier free formula such that the given formula is equivalent to the quantifier free formula for all k-good sets; player II is trying to show that this is not the case.

The game proceeds in rounds. At the start of any round, player I has the following options. Let q be the current main formula at the start of the round.

1) If the formula q is quantifier-free, the game is over, and player I has won.
2) Player I may elect to write down an integer k and a new formula q'. If for all k-good sets S and any assignment of values in $\mathbf{Z}$ to the variables, the two formulae are

equivalent, then the round is completed and q' is the next "current main formula". If not, player I immediately loses this play of G,

3) The next option is available to player I if the current main formula has the form $\gamma \wedge \delta$ where γ is quantifier-free. He simply writes down δ as the next "current main formula" and the round is over.

4) The final option has player I demanding a response from player II. This option is available if the current main formula q is a disjunction $\theta_1 \vee \cdots \vee \theta_k$. (Player I will explicitly present a particular representation of q as a disjunction to player II.) Player II will then choose one of the disjuncts θ_i to be the next "current main formula" ending this round.

If a play of the game does not terminate at any finite stage, then player II is declared the winner.

LEMMA 4. If player I has an effective winning strategy for the game G then Lemma 3 is true.

Proof:

We fix a recursive winning strategy σ for player I in the game G. Using σ we define an auxiliary procedure P. Our intention is that P takes as input a position p in the game G in which player I has played according to σ, and supplies an integer k and a quantifier-free formula q' such that q' is equivalent to the current main formula of position p for all k-good sets S.

After we describe the algorithm for P we will show that P works as intended by contradiction. That is, on the assumption that P does not "function correctly" we construct a play of the game G in which player II succeeds in defeating player I's winning strategy σ. Thus procedure P will be shown to "function correctly". Using P, we will easily construct a suitable procedure $\mathcal{L}$ witnessing the truth of Lemma 3. This will be easy since player I eventually produces a k and a quantifier free formula.

We view the position p as recording the entire history of the game up to that point including which option I chose at each prior round, and the sequence of "main formulae" as well as the choices of player II.

Here is a description of the procedure P. If the current main formula q is quantifier-free, then P returns the value 1 for k and returns q as its quantifier-free answer.

Next suppose that player I, using strategy σ, will invoke the option 2 writing down a number k' and a formula q'. Call the resulting position p'. Procedure P calls upon itself with the input p'. Let us suppose the returned values are k'' and q''. (It could happen, of course, that the call to procedure P with input p' never terminates. In that case, the call with input p will also never terminate.) Then P returns $k := \max(k', k'')$ and q'' as its number and formula.

Next suppose that player I chooses option 3. So q has the form $\gamma \wedge \theta$ where γ is quantifier-free, and for the succeeding position p', the current main formula will be θ. Procedure P calls upon itself with the input p' getting the answers k and θ'. P then returns k and $\gamma \wedge \theta'$.

Finally suppose that player I chooses option 4. In this case he specifies a decomposition of the current main formula as a disjunction $\theta_1 \vee \cdots \vee \theta_n$. Player II gets to choose one of the θ_i's as the next main formula. Let p_i be the position resulting after player II's choice of θ_i. Then procedure P calls itself with the successive inputs $p_1, \ldots, p_n$ getting the answers k_i and θ'_i to its call with input p_i. Procedure P then returns $k := \max\{k_1, \ldots, k_n\}$ and $q' := \theta'_1 \vee \cdots \vee \theta'_n$.

This completes our description of procedure P. We next demonstrate that for any initial position (i.e, the position immediately resulting from player II's writing down a formula $\exists x\theta$ with θ quantifier-free) procedure P returns a correct answer.

Suppose not toward a contradiction. Then one easily constructs a sequence $\{p_n\}_{n\geq 1}$ of positions such that:

1) p_n is the position at the beginning of the n^{th} round of a play of G in which I plays according to σ.
2) P does not return a correct answer on input p_n.

It follows first, the current main formula in this play of G is never a quantifier-free formula. (Else P would give a correct answer on input the corresponding position.) Also the play goes on forever. So II wins. But this contradicts our assumption that σ is a winning strategy for player I.

But now it is easy to construct the desired procedure $\mathcal{L}$. On input q of the form $\exists x\theta$, $\mathcal{L}$ constructs the beginning position p in the game G that corresponds to q. Then $\mathcal{L}$ calls upon P with input p and gets the answers k and q'. It then itself returns k and q' as its own answers. This completes the proof of Lemma 4. $\boxtimes$

We return to the proof of Lemma 3. We shall describe a winning strategy for I in the game G just described. (Of course, once Lemma 3 is proved, player I has a trivial winning strategy employing $\mathcal{L}$.) We shall let the game metaphor and player II recede into the background. After supplying the initial formula, player II's role is only to pick, from time to time, one of the disjuncts of the current main formula. We shall say instead that player I *focuses on a disjunct*. This should cause no confusion if it is firmly understood that player I does not himself choose which disjunct will be focused on.

It would be tedious and uninformative to keep track in our notation of the many different formulae that will occur in the course of the transformation from our initial main formula q to the final quantifier-free formula q'. Thus we speak of performing a transformation on the formula θ rather than of replacing the formula θ by some other formula θ' when we think no loss of clarity results.

We turn now to the details of the description of the winning strategy σ for player I needed to complete the proof of Lemma 3.

Step 1:

Our initial formula q has the form $\exists x\theta$, where θ is a quantifier-free formula. Our first step is designed to set things up so that we can apply the usual Presburger elimination algorithm (see [3] for a presentation of this algorithm). In order to do this, we must separate those terms involving $\mathcal{A}$, $\lfloor\cdot\rfloor$, and $\lceil\cdot\rceil$ from the rest of the formula.

First, we replace every sub-formula of the form $\mathcal{A}(t')$ with the formula $\lceil t'\rceil = t'$. Next we eliminate terms t of the form $\lfloor t'\rfloor$ or $\lceil t'\rceil$. Intuitively, notice that the formula $t = \lfloor t'\rfloor$ means $t \in S$, $t \leq t'$, and no element of S lies between $t+1$ and t'. Informally, we eliminate t by explicitly defining two new existentially quantified variables z_{2i} and z_{2i+1} such that $z_{2i} \leq t' < z_{2i+1}$, and z_{2i} and z_{2i+1} are adjacent elements of S; then replace t by z_{2i}. The formula $t = \lceil t'\rceil$ is handled similarly, except that at the end, t is replaced by z_{2i+1}. The above description is not quite accurate because we must also cover the case where no element of S is less than or equal to t', so that $\lfloor t' \rfloor = 0$; and because we must make an extra effort to separate the part of the modified formula involving S from the rest of the formula. Formally, we proceed as follows. (It is important to specify carefully the order in which we tackle the terms t and to prove that the procedure converges. We shall handle these points in a moment.)

For every term t of the form $\lfloor t' \rfloor$ or $\lceil t' \rceil$ we perform the following steps:

1) Introduce new (existentially quantified) variables z_{2i} and z_{2i+1} (where i is the number of floor and/or ceiling terms that have been previously processed).
2) Conjoin a formula ψ_i stating that z_{2i} and z_{2i+1} represent adjacent elements of S, i.e.,

$$(\mathcal{A}(z_{2i}) \vee (z_{2i} = 0)) \wedge (z_{2i+1} = \lceil z_{2i} + 1 \rceil).$$

The formula ψ_i is not considered to be part of the new θ.

3) Replace t with the appropriate new variable, and conjoin to θ a formula that states that z_{2i} and z_{2i+1} surround t'. That is, if t is $\lfloor t' \rfloor$, then we perform the transformation:

$$\theta := (((z_{2i} \leq t') \vee (z_{2i} = 0)) \wedge (t' < z_{2i+1}) \wedge \theta[z_{2i}/t]).$$

If t is $\lceil t' \rceil$, then we perform the transformation:

$$\theta := (((z_{2i} < t') \vee (z_{2i} = 0)) \wedge (t' \leq z_{2i+1}) \wedge \theta[z_{2i+1}/t]).$$

Note that special care is taken to allow $(z_{2i} = 0)$ in case we have $\lfloor t' \rfloor = 0$ (for $t = \lfloor t' \rfloor$) or $\lfloor t' - 1 \rfloor = 0$ (for $t = \lceil t' \rceil$).

We now spell out the order in which the terms t are to be processed. Call the *box rank* of a term t the number of occurrences of the floor and ceiling function symbols in t. We always pick to be processed a term t of maximal box rank in the current version of θ. Suppose that this term has box rank r. Inspection of the step that eliminates this term will show that no new terms of box rank greater than r are introduced, and that the number of terms of box rank r is actually decreased. Thus as our procedure runs the terms of box rank r and above will eventually all be eliminated. Then all the terms of box rank $r - 1$ will all be eliminated, and so forth. Eventually, the procedure will terminate in a formula θ not containing any floor or ceiling function symbols.

At the end of this procedure, the entire formula has the form:

$$\exists z_0 \exists z_1 \cdots \exists z_{2n} \exists z_{2n+1} \exists x \, [\psi_0 \wedge \cdots \wedge \psi_n \wedge \theta]$$

We define ψ to be $\psi_0 \wedge \cdots \wedge \psi_n$, and write the formula as $\exists \vec{z} \exists x \, [\psi \wedge \theta]$.

It will be convenient later to know that the set of variables $\vec{z}$ is non-empty. We can always arrange this by replacing, if necessary, the formula θ by $\theta \wedge (0 = \lfloor 0 \rfloor)$ before commencing Step 1.

Step 2:

Since the variable x does not appear in the formula ψ, our current formula is equivalent to the formula $\exists \vec{z} \, [\psi \wedge \exists x \, \theta]$. Moreover, we have arranged matters so that neither the predicate $\mathcal{A}$ nor the function symbols $\lfloor \cdot \rfloor$ or $\lceil \cdot \rceil$ appear in θ. Hence we can apply the standard quantifier elimination procedure for Presburger arithmetic to replace $\exists x \, \theta$ by a quantifier-free L_1-formula (which we again call θ).

Step 3:

Our next task is to transform the formula θ so that it remains an L_1-formula, but contains none of the predicates D_j. For the transformations of the current step to be valid, we must require that $k \geq j$ for any j such that D_j occurs in θ. It follows that if S is k-good, and $z \in S \cup \{0\}$ then $z \equiv 0 \bmod j$.

Let $(\exists \vec{z})[\psi \wedge \theta]$ be our current "main formula". The formula θ is a quantifier-free L_1-formula. We put it in disjunctive normal form. Thus θ is logically equivalent to a disjunction $\theta^1 \vee \cdots \vee \theta^m$ where each of the θ^j is a conjunction of atomic L_1-formulae and negations of atomic L_1-formulae.

It follows that the main formula is logically equivalent to the disjunction:

$$(\exists \vec{z})[\psi \wedge \theta^1] \vee \cdots \vee (\exists \vec{z})[\psi \wedge \theta^m] \qquad (\star)$$

We shift our focus to one of the disjuncts of $(\star)$ and change our notation so that θ refers to one of the formulae θ^r. What we have gained by this is that θ is now a conjunction of atomic formulae and negations of atomic formulae of L_1.

Let us call two L_1-terms equivalent if they take the same values in $\mathbf{Z}$ for every assignment of values in $\mathbf{Z}$ to the variables of L_1. Clearly up to equivalence, every term can be represented as a linear combination $a_0 + a_1 \cdot v_1 + \cdots + a_p \cdot v_p$ where $v_1, \cdots, v_p$ are distinct variables of L_1 and the coefficients a_i are elements of $\mathbf{Z}$. We shall speak loosely from here on out and act as if the L_1-terms are such linear combinations (though clearly this is not literally true).

Consider a conjunct of θ in which some D_j appears. The requirements we have imposed on k together with the formula ψ insure that the values of the variables of $\vec{z}$ must be chosen $\equiv 0 \bmod j$. Thus we can find an equivalent formula to this conjunct in which none of the variables of $\vec{z}$ appears. But then we can move the conjunction of all the conjuncts of θ in which a D_j appears past the block of quantifiers $\exists \vec{z}$. The resulting formula has the form:

$$\gamma \wedge (\exists \vec{z})[\psi \wedge \theta]$$

where γ is quantifier-free and now θ is a quantifier-free L_1-formula containing none of the predicates D_j.

For the remainder of our discussion, we can focus our attention on the second conjunct of this formula. The net result is that our current formula now has the form $(\exists \vec{z})[\psi \wedge \theta]$ where θ is a quantifier-free L_1-formula not containing any of the D_j predicates.

Step 4:

Before beginning this step, we have to handle a technical point. A finite conjunction of L^{e}-formulae can be rendered in several tautologically equivalent forms because of the different possibilities for ordering and grouping the conjuncts. The reader should fix on some convention to eliminate this ambiguity (which we shall ignore in the future). One approach that works is to always use the rendering of minimal Gödel number.

Let $\chi_1, \chi_2, \ldots, \chi_m$ be all possible orderings of the $\vec{z}$ variables, including their relation to 0. Examples of such orderings are $0 = z_0 < z_2 = z_1 < z_3$ and $0 < z_2 < z_3 < z_0 < z_1$. Only orderings which are consistent with ψ should be included. For example, any ordering with $z_2 = z_3$ is impossible (since $z_3 = \lceil z_2 + 1 \rceil$), as is the ordering $0 < z_0 < z_2 < z_1 < z_3$, since no element of the k-good set can occur strictly between z_0 and z_1.

Here is a precise statement of the conditions we impose on the χ_j's:

1) Let T be the set of all terms that are either 0 or one of the variables of $\vec{z}$. Then χ_j

will be a conjunction of formulae of the forms $t < t'$ and $t = t'$ where t and t' are members of T.

2) Let t and t' be elements of T. Then exactly one of the formulae $t < t'$, $t = t'$, and $t' < t$ is a conjunct of χ_j.
3) $(\exists \vec{z})\chi_j$ is consistent with Presburger arithmetic.
4) Let z_{2i} and z_{2i+1} be variables of $\vec{z}$. Then the formula $z_{2i} < z_{2i+1}$ is a conjunct of χ_j. Moreover, for no $t \in T$ are the formulae $z_{2i} < t$ and $t < z_{2i+1}$ both conjuncts of χ_j.

It is clear that we can effectively generate the list of formulae $\chi_1, \ldots, \chi_m$ satisfying the conditions just enumerated.

The formula $(\exists \vec{z})[\psi \wedge \theta]$ is equivalent to

$$\bigvee_{j=1}^{m} (\exists \vec{z})[\psi \wedge \theta \wedge \chi_j] \qquad (\star\star)$$

We focus our attention on one of the disjuncts of $(\star\star)$. We shift notation and denote the formula χ_j by χ. The effect of this transformation is that we now have complete information on the ordering of the terms in T.

Step 5:

Our next step is to reorganize the information contained in $\psi \wedge \chi$ into a particularly transparent form. The formula χ defines an equivalence relation on T as follows: terms t and t' are equivalent iff "$t = t'$" is a conjunct of χ. Since we have arranged matters so that T contains at least the variables z_0 and z_1, it is clear that T has at least two equivalence classes. We order the equivalence classes of T in the evident way: The equivalence class of t is less than the equivalence class of t' iff the formula χ contains a conjunct "$t < t'$".

Let the number of equivalence classes of T be $r + 1$. We have already remarked that $r > 0$. Let $w_1, \ldots w_r$ be fresh variables (not appearing in our current main formula). Let W be the set of terms $\{0, w_1, \ldots, w_r\}$.

To each term of T we associate a term of W as follows: If "$t = 0$" is a conjunct of χ, then the term associated to t is 0. Otherwise, let the equivalence class of t be the i^{th} largest equivalence classes of T. Then we associate the term w_i to t.

Let $\psi^\star$ be the conjunction of the following formulae:

1) For $1 \leq i \leq r-1$, the formula "$w_i > w_{i+1}$".

2) The formula "$w_r > 0$".

3) The formula "$\mathcal{A}(w_i)$" (for each i such that $1 \leq i \leq r$).

4) Let z_{2i} be one of the $\vec{z}$ variables. Let $w \in W$ be the term associated to z_{2i} and let $w' \in W$ be the term associated to z_{2i+1}. Then the formula "$w' = \lceil w+1 \rceil$" will be a conjunct of $\psi^\star$.

This completes our description of $\psi^\star$.

Let $\theta^\star$ be obtained from θ by replacing each variable z_j in θ by the associated term in W. Then it should be clear that the formula $(\exists \vec{z})[\psi \wedge \chi \wedge \theta]$ is equivalent to the formula $(\exists \vec{w})[\psi^\star \wedge \theta^\star]$.

We make two cosmetic changes to complete this step. First we replace each of the variables w_i by the variable z_i. Second, we drop the $\star$ from our notation for $\theta^\star$ and instead call the resulting formula θ. Note that it is no longer the case that z_{2i} and z_{2i+1} are necessarily adjacent.

Thus at the end of this step, our current formula has the form: $(\exists \vec{z})[\psi^\star \wedge \theta]$. θ is a quantifier-free L_1-formula, which is a conjunction of formulae that are either atomic or the negations of atomic formulae. Moreover, none of the predicates D_j appear in θ.

Step 6:

This step is a relatively minor one. We wish to organize the formulae in θ in a slightly more useful form.

Recall that we are viewing L_1-terms as linear combinations of the variables and the constant 1 with integral coefficients. We replace the various atomic formulae of θ by equivalent atomic formulae so that the left-hand side of each new atomic formula is either 0 or a linear combination of the variables in $\vec{z}$, while no variable from $\vec{z}$ appears on the right hand side of any new atomic formula. (It may be necessary to multiply an inequality by -1 to achieve this migration of the $\vec{z}$'s to the left hand side, since $>$ is not a basic predicate of our language.)

We sum up what we have achieved in this step. We may assume that θ is a Boolean combination of atomic formulae whose left-hand side is either 0 or a non-zero linear combination of the variables from $\vec{z}$ and whose right-hand side contains no variables from $\vec{z}$.

Each of these formulae has as its predicate one of the symbols $<$ or $=$.

Step 7:

We will produce a formula θ', and a value of k, such that for all k good sets, $(\exists\vec{z})[\psi^\star \wedge \theta] \equiv (\exists\vec{z})[\psi^\star \wedge \theta']$ and θ' is a quantifier-free L^{e}-formula of a very simple form: every atomic formula in θ' either contains none of the $\vec{z}$ variables or is of one of the forms: $z_i < t$, $t < z_i$, or $z_i = t$, where t does not involve any of the $\vec{z}$ variables.

For the course of the processing involved in this step we introduce a new predicate $>$ to the language L^{e}. (We have the obvious semantics for this new predicate: "$x > y$" is equivalent to "$y < x$".) This will make it possible for us to maintain our desire that the $\vec{z}$ variables remain on the left of our atomic formulae. At the end of this step we will eliminate $>$ (in favor of $<$) in the obvious way.

The following are the inductive assumptions that we will preserve about θ during the course of this step.

1) θ is a Boolean combination of atomic formulae.
2) Let α be one of the atomic formulae appearing in θ. Then the predicate of α is one of $<$, $=$, or $>$.
3) Let α be as above. The left-hand side of α will be either 0 or a linear combination of the variables $\vec{z}$. The right-hand side of α contains no variables from $\vec{z}$. (It may however contain the function symbols $\lfloor\cdot\rfloor$ or $\lceil\cdot\rceil$.)

Define the rank of an atomic formula meeting the conditions imposed on α above as follows. Let the left-hand side of α be equivalent to a linear combination $\sum_{j=1}^{r} a_j \cdot z_{n_j}$ where the a_j's are non-zero, and $n_1 < \cdots < n_r$. (r could be zero, in which case this sum is just 0.) If $r > 1$, we say the rank of α is r. If $r = 1$ and $a_1 \neq 1$, we say the rank of α is 1. Otherwise, we say the rank of α is 0.

We shall describe a transformation that achieves the following. If the maximal rank of any atomic formula appearing in θ is $r > 0$ and the number of atomic formulae of rank r is s, then after the transformation, the maximum rank will be $\leq r$, and the number of atomic formulae of rank r will be $< s$. Clearly by iterating such transformations (always tackling an atomic formula of maximal rank) we will eventually transform θ so that it has no atomic formulae of rank > 0, thereby completing the goal of this step.

So we turn our attention to an atomic formula, α, of θ of maximal positive rank. To ease the notation, we shall consider the case when $n_j = j$ for $1 \le j \le r$. The general case is handled similarly.

Keep in mind that, for this transformation, formulae are called equivalent if they are equivalent under the assumption that the ordering on the z_i's is that described in $\psi^\star$. We will find a formula β that is equivalent to α (for k-good sets for an appropriate value of k), but where every atomic formula in β has summands with fewer than r terms. Applying this recursively, we eventually get a formula, all of whose atomic subformulae are of the desired form. As it happens, we can also apply case 1 of the transformation when $r = 1$ to get rid of the constant multiplier in single term summands.

We may assume that $a_1 > 0$. (If necessary, multiply both sides of α by -1.) To insure the validity of the transformations of this step, we impose the requirement that k be greater than $\sum_i |a_i|$.

The intuition about the transformations is that since (1) S is a k-good set, (2) for all i $z_i \in S$, and (3) $z_1 > z_2 > \cdots > z_r$, we know that z_1 is much bigger than the other z-terms; hence its relation to t should be the only important factor in determining (say) whether $\sum_i a_i z_i < t$. This will allow us to (almost) get rid of all the other variables. The following calculations help to pin down this intuition.

$$\begin{aligned}
\sum_i a_i z_i &= (a_1 - 1)z_1 + z_1 + \sum_{i \ge 2} a_i z_i \\
&\ge (a_1 - 1)z_1 + z_1 - \sum_{i \ge 2} |a_i| z_2 \\
&> (a_1 - 1)z_1 + k z_2 - \sum_{i \ge 2} |a_i| z_2 \quad (\text{since } k z_2 < z_1) \\
&\ge (a_1 - 1)z_1 + (k - \sum_{i \ge 2} |a_i|) z_2 \\
&\ge (a_1 - 1)z_1 \quad (\text{since } k > \textstyle\sum_i |a_i|)
\end{aligned}$$

$$\begin{aligned}
\sum_i a_i z_i &\le a_1 z_1 + \sum_{i \ge 2} |a_i| z_i \\
&\le a_1 z_1 + k z_2 \quad (\text{since } \textstyle\sum_i |a_i| < k) \\
&< (a_1 + 1) z_1 \quad (\text{since } k z_2 < z_1)
\end{aligned}$$

Combining the above calculations, one obtains

$$(a_1 - 1)z_1 < \sum_i a_i z_i < (a_1 + 1)z_1$$

We now exhibit a concrete example of the z_1 term determining the relation between $\sum_i a_i z_i$ and t.

$$\begin{aligned} z_1 < \lfloor t \rfloor &\Rightarrow k z_1 < \lfloor t \rfloor \quad (\text{since } z_1 \text{ and } \lfloor t \rfloor \text{ are both in } S) \\ &\Rightarrow (a_1 + 1) z_i < \lfloor t \rfloor \quad (\text{since } a_1 < k\ ) \\ &\Rightarrow \sum_i a_i z_i < \lfloor t \rfloor \quad (\text{since } \textstyle\sum_i a_i z_i \leq (a_1 + 1) z_i) \\ &\Rightarrow \sum_i a_i z_i < t \quad (\lfloor t \rfloor \neq 0 \text{ since } z_1 < \lfloor t \rfloor) \end{aligned}$$

We now proceed to transform every atomic formula. There are three cases, depending on the magnitudes of a_1 and a_2. Within each case we must deal with three types of atomic formulae, namely $\sum_i a_i z_i < t$, $\sum_i a_i z_i = t$, and $\sum_i a_i z_i > t$. We examine one type of atomic formula under one case in detail, and then state what happens for the other cases and atomic formulae. The omitted details consist of calculations similar to the above.

Case 1: $(a_1 > 1)$

Consider the atomic formula $\sum_i a_i z_i < t$. It is true iff the following formula is true:

$$\left((z_1 < \lfloor t \rfloor) \wedge (\sum_i a_i z_i < t)\right) \vee \left((z_1 = \lfloor t \rfloor) \wedge (a_1 \lfloor t \rfloor + \sum_{i \geq 2} a_i z_i < t)\right) \vee \left((z_1 > \lfloor t \rfloor) \wedge (\sum_i a_i z_i < t)\right).$$

Since $z_1 < \lfloor t \rfloor \Rightarrow \sum_i a_i z_i < t$, the first disjunct is equivalent to $z_1 < \lfloor t \rfloor$. Using $(a_1 - 1)z_1 < \sum_i a_i z_i$ and $a_1 > 1$, we get $\sum_i a_i z_i < t \Rightarrow (a_1 - 1)z_1 < t \Rightarrow z_1 < t \Rightarrow z_1 \leq \lfloor t \rfloor$. (The last implication follows from $z_1 \in S$ and the definition of $\lfloor \cdot \rfloor$.) This implies that the third disjunct is identically false. Therefore the original formula is equivalent to

$$(z_1 < \lfloor t \rfloor) \vee \left((z_1 = \lfloor t \rfloor) \wedge \left(a_1 \lfloor t \rfloor + \sum_{i \geq 2} a_i z_i < t\right)\right).$$

In turn, this is equivalent to

$$(z_1 < \lfloor t \rfloor) \vee \left((z_1 = \lfloor t \rfloor) \wedge \left(\sum_{i \geq 2} a_i z_i < t - a_1 \lfloor t \rfloor\right)\right).$$

This is of the desired form. In the other cases discussed we leave to the reader the massaging needed to bring the non-$\vec{z}$ terms to the right.

Thus, the atomic formula $\sum_i a_i z_i = t$ is equivalent to

$$(z_1 = \lfloor t \rfloor) \wedge \left(a_1 \lfloor t \rfloor + \sum_{i \geq 2} a_i z_i = t \right).$$

Likewise, the atomic formula $\sum_i a_i z_i > t$ is equivalent to

$$(z_1 > t) \vee \left((z_1 = \lfloor t \rfloor) \wedge \left(a_1 \lfloor t \rfloor + \sum_{i \geq 2} a_i z_i > t \right) \right).$$

Case 2: $(a_1 = 1)$ and $(a_2 > 0)$. All subcases have identical outcomes to those of Case 1.

Case 3: $(a_1 = 1)$ and $(a_2 < 0)$.

$\sum_i a_i z_i < t$ is equivalent to

$$(z_1 < t) \vee \left((z_1 = \lceil t \rceil) \wedge \left(a_1 \lceil t \rceil + \sum_{i \geq 2} a_i z_i < t \right) \right).$$

$\sum_i a_i z_i = t$ is equivalent to

$$(z_1 = \lceil t \rceil) \wedge \left(a_1 \lceil t \rceil + \sum_{i \geq 2} a_i z_i = t \right).$$

$\sum_i a_i z_i > t$ is equivalent to

$$(z_1 > \lceil t \rceil) \vee \left((z_1 = \lceil t \rceil) \wedge \left(a_1 \lceil t \rceil + \sum_{i \geq 2} a_i z_i > t \right) \right).$$

Step 8:

Transform θ into disjunctive normal form. Now replace each negation of an atomic formula by a disjunction of atomic formulae. (For example, replace $x \neq y$ by $x < y \vee y < x$.) Put θ into disjunctive normal form again. Thus at this point θ is a disjunction of conjunctions of atomic formulae.

Now move the disjunctions out past the existential quantifiers and focus on one of the disjuncts. The effect is that we are left with a θ that is a conjunction of atomic formulae.

Moreover, each of these formulae either contains none of the $\vec{z}$ variables, or has a single $\vec{z}$ variable occurring as the term on one of its sides.

Step 9:

We say that t is a high level term of θ if:

1) t appears on the left or right side of one of the conjuncts of θ;
2) None of the variables of $\vec{z}$ appear in t.

We next arrange that every high-level term in θ is a floor or ceiling, as follows. Replace every atomic formula in θ of the form $z_i < t$ with $z_i < \lceil t \rceil$. Replace every atomic formula in θ of the form $z_i > t$ with $z_i > \lfloor t \rfloor$. For every atomic formula in θ of the form $z_i = t$, replace it with $z_i = \lfloor t \rfloor$, and add the conjunct $t = \lfloor t \rfloor$ outside the quantifier block. All these transformations are valid via $z_i \in S$ and the definitions of $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$. (E.g. if $z_i < t$, then $t \leq \lceil t \rceil$ implies $z_i < \lceil t \rceil$. Conversely, if $z_i < \lceil t \rceil$ then $z_i < t$, since $t \leq z_i$ and $z_i \in S$ implies $\lceil t \rceil \leq z_i$, a contradiction.)

After this is done, our main formula has the form $\gamma \wedge (\exists \vec{z})[\psi^\star \wedge \theta]$ where γ is quantifier-free and contains no variables from $\vec{z}$. We focus on the second conjunct of the main formula. This achieves our goal that every high level term of θ is either a floor or a ceiling.

Let U be a non-empty set of L^{e} terms and χ a formula of L^{e}. We say that χ *gives an ordering of* U if the following conditions are met:

1) χ is the conjunction of a set of atomic sentences.
2) Let α be a conjunct of χ. Then α has the form sRt where s and t are members of U and R is one of the relation symbols "$=$" and "$<$".
3) There is a surjective map $f : U \to k$ (where k is a non-negative integer and hence is equal to the set of integers less than k) such that for s, t elements of U: (a) "$s < t$" is a conjunct of χ iff $f(s) < f(t)$; (b) "$s = t$" is a conjunct of χ iff $f(s) = f(t)$.

It is clear that we can effectively determine, given U, the finite set of χ's that give an ordering of U.

We let H be the set of high level terms of θ together with terms for 0 and the least element of S, namely $\lceil 0 + 1 \rceil$. Let $V = \{z_1, \ldots, z_r\}$.

Let χ be a formula that gives an ordering of $V \cup H$. We say that χ is compatible with θ if every conjunct of θ is a conjunct of χ.

Let χ be as above. We say that χ is compatible with ψ^* if the following conditions hold:

1) Whenever a formula of the form sRt is a conjunct of $\psi^\star$ (where s, $t \in (V \cup H)$ and $R \in \{=, <\}$) then sRt is a conjunct of χ.
2) Suppose that $t = \lceil s+1 \rceil$ is a conjunct of $\psi^\star$. Then for no term $u \in (V \cup H)$ are $s < u$ and $u < t$ both conjuncts of χ.

Let $\chi_1, \ldots, \chi_s$ be those formulae giving orderings of $V \cup H$ which are compatible with θ and $\psi^\star$. Then it is clear that the formula $(\exists \vec{z})[\psi^\star \wedge \theta]$ is equivalent (for any 1-good set S) to the formula:

$$\bigvee_{j=1}^{s} (\exists \vec{z})[\psi^\star \wedge \chi_j] \tag{$\star\star\star$}$$

It could conceivably happen that $s = 0$. I.e., there are no orderings of $V \cup H$ compatible with θ and $\psi^\star$. In that case, the current main formula is easily seen to be equivalent (for all 1-good sets S) to the quantifier-free formula "0=1" and we are done.

We focus on one of the disjuncts of $(\star\star\star)$ and refer to the corresponding χ_j as χ.

Step 10:

We are finally ready to eliminate the quantifiers from our current main formula.

Step 10 proceeds in several stages, and we first describe the inductive assumptions that we will maintain during this process.

1) H is a set of terms each of which has one of the forms $\lceil t \rceil$ or $\lfloor t \rfloor$ for some term t in which none of the variables $\vec{z}$ appear. (As Step 10 proceeds we will add terms to H but never remove terms.)
2) V is a subset of $\{z_1, \ldots, z_r\}$ (As Step 10 proceeds, variables are deleted from V but never added.)
3) χ gives an ordering of $H \cup V$.
4) $\psi^{\star\star}$ is a conjunction of atomic formulae whose conjuncts are precisely those clauses of $\psi^\star$ in which only z_i's from V appear.
5) The current main formula is $(\exists \vec{V})[\psi^{\star\star} \wedge \chi]$ where $\vec{V}$ is the elements of V in some order.
6) Let s and t be terms such that $t = \lceil s+1 \rceil$ is a clause of $\psi^{\star\star}$. Then for no term $u \in H$ are the formulae $s < u$ and $u < t$ both conjuncts of χ.

At the start of this step, H, V and χ have the values they had at the end of Step 9. $\psi^{\star\star}$ is the formula $\psi^{\star}$. It is clear that all our inductive assumptions are met.

We indicate various processes which we can sometimes apply. When one of the processes applies, it removes some variables from V, maintains our inductive assumptions and replaces the old main formula by one which is equivalent (for any 1-good set S). We shall show that one of these processes applies unless the set V is empty. But when V is empty we have succeeded in converting the current main formula to a quantifier free formula, and we are done with the quantifier elimination.

Case 1: For some term $t \in H$ and some variable $v \in V$, χ contains the conjunct $t = v$.

In this case, the basic idea in eliminating v is clear. We will delete all the clauses in χ involving v. This almost works but there is one extra wrinkle.

The variable v may appear in $\psi^{\star\star}$ in clauses of the form "$y_1 = \lceil y_2 + 1 \rceil$ (where $y_1, y_2 \in V \cup \{0\}$, and one of them is v.) In that case we proceed as follows.

If, for some term s, the formula $s = \lceil v + 1 \rceil$ is a conjunct of $\psi^{\star\star}$, add the term $\lceil t + 1 \rceil$ to H. Add the clause $\lceil t + 1 \rceil = s$ to χ. (Note that from the definition of $\psi^{\star}$ there is at most one such s.)

There is at most one way of prolonging χ to give an ordering of the "new" $V \cup H$. (Since we have added a single term to the set being ordered and required that the new ordering make it equal to the old element s.) If there is no such way (since $\lceil t + 1 \rceil$ was already in H, and the old χ made $s = \lceil t + 1 \rceil$ false), then the current main formula is clearly equivalent to $0 = 1$ and we are done.

If not, we continue. There might be a formula of the form $v = \lceil s + 1 \rceil$ which is a conjunct of $\psi^{\star\star}$. (Again, this is true for at most one term s.) If so, add the term $\lfloor t - 1 \rfloor$ to H and the clause $\lfloor t - 1 \rfloor = s$ to χ. Again prolong χ to give an ordering of $H \cup V$, if possible. If it is impossible to so prolong χ, the current main formula will be equivalent to "$0 = 1$" and we will be done with the quantifier elimination process.

The justification for the maneuver of the proceding paragraph is as follows. Let x and y be elements of $S \cup \{0\}$ with $y > 0$. Then the formulae $y = \lceil x + 1 \rceil$ and $\lfloor y - 1 \rfloor = x$ both say that x and y are consecutive members of $S \cup \{0\}$ with $x < y$.

Finally we delete v from V, and redefine $\psi^{\star\star}$ so as to meet clause 4 of our inductive requirements. We leave to the reader to verify that if we do this all our inductive requirements are met, and that the new main formula is equivalent to the previous one.

Case 2: Case 1 does not hold but there are consecutive terms a and b of H (with respect to the ordering of H given by χ) such that for some variable $v \in V$, χ decrees that $a < v$ and $v < b$.

In that case, fix such an a and b and let $W = \{v \in V \,|\, \chi$ decrees that $a < v$ and $v < b\}$.

We claim that if the clause $t = \lceil u + 1 \rceil$ appears in $\psi^{\star\star}$ and one of u and t appears in W, then both u and t appear in W. There are several cases to consider in seeing this. First, we cannot have u the term 0 and $t \in W$, since then Case 1 would apply. (Recall that $\lceil 0 + 1 \rceil \in H$.) Similarly, since Case 1 does not apply, χ does not decree that $u = a$ or that $b = t$. So if, for example, $t \in W$ and $u \notin W$, then we must have χ decreeing that $u < a$ and $a < t$. But this contradicts clause 6 of our inductive assumptions.

We let s be the cardinality of W. For $0 \le i \le s$ we define a term a_i, by induction on i, as follows. The term a_0 is just a; a_{i+1} shall be $\lceil a_i + 1 \rceil$.

Let $H' = H \cup \{a_s\}$. Let V' be $V - W$. Let χ' be the formula that orders $V' \cup H'$ by an ordering that prolongs the ordering on $V' \cup H$ given by χ and which places a_s between a and b. There is at most one χ' meeting these requirements since a and b are adjacent elements of $V' \cup H$ in the ordering given by χ. The only way there could fail to be such a χ' is if $a_s \in H$. In that case, it will result from the following discussion that the current main formula is equivalent to $0 = 1$ (since χ decrees that $b \le a_s$) and we will be done.

We define ψ' to be the conjunction of those clauses of $\psi^\star$ which only contain variables from V'. Then for any 1-good S we have:

$$(\exists \vec{V})[\psi^{\star\star} \wedge \chi] \Leftrightarrow (\exists \vec{V}')[\psi' \wedge \chi'] \qquad (\star 4)$$

Let $\{w_1, \ldots, w_s\}$ be the members of W arranged in increasing order. (I.e., for $1 \le i < j \le s$, $w_i < w_j$ is a conjunct of $\psi^\star$.)

We first argue the direction from left to right in ($\star$4). Let an interpretation be given to the variables appearing in $\psi^{\star\star} \wedge \chi$. We have to argue that $\psi' \wedge \chi'$ is true as well. For ψ' this is trivial since every conjunct of ψ' is a conjunct of $\psi^{\star\star}$ as well. From the way that

χ' was defined, it will be true provided that $a_s < b$. In turn, from the definition of a_s this will be true iff there are s elements of S between a and b. But χ guarantees that there are; they are given by the values of $w_1, \ldots, w_s$.

Next we argue the direction from right to left in $(\star 4)$. Let an interpretation be given to the variables appearing in $\psi' \wedge \chi'$. We show how to prolong this interpretation so as to make $\psi^{\star\star} \wedge \chi$ true. For $1 \leq i \leq s$, we assign to the variable w_i the value of the term a_i. We need to know that $a < w_1 < \cdots < w_s < b$. But this is insured by the clause $a_s < b$ of χ'.

We have to see that all the clauses of $\psi^{\star\star}$ come out true. The only problematical ones are those that do not appear also in χ or in ψ'. These have the form $v = \lceil u + 1 \rceil$ where at least one of u, v is in W. If both u and v are in W, this clause will come out true (since $a_{j+1} = \lceil a_j + 1 \rceil$ for $1 \leq j < s$.) u cannot be the term 0, since we threw $\lceil 0 + 1 \rceil$ into H, and Case 1 does not apply. If, for example, $v \notin W$ and $u \in W$, we would have $u < b < v$ contradicting clause 6 of our inductive assumptions. The remaining case (that $u \notin W$, $u \in V$, and $w \in W$ is dismissed similarly.) This completes the proof of the right to left direction of $(\star 4)$.

But now we simply set $\psi^{\star\star} := \psi'$, $\chi := \chi'$, $H := H'$ and $V := V'$. We take as our new main formula $(\exists \vec{V})[\psi^{\star\star} \wedge \chi]$. It is clear from the proceding that the new main formula is equivalent to the old one, that the cardinality of V has decreased and that our inductive assumptions have been maintained. This completes the discussion of Case 2.

Case 3: Cases 1 and 2 do not obtain, but the set V is non-empty.

The treatment of this case is similar to that of Case 2, but easier. Let t be the largest element of H with respect to the ordering given by χ. (H is certainly non-empty since $0 \in H$.) Then χ must put all the elements of V larger than t. Let the cardinality of V be s, and let the elements of V in increasing order be $w_1, \ldots, w_s$.

Let χ' be the formula obtained from χ by dropping all the conjuncts which mention variables of V. Then for any 1-good set S, we have:

$$(\exists \vec{V})[\psi^{\star\star} \wedge \chi] \Leftrightarrow \chi' \qquad (\star 5)$$

Of course, the left to right implication of ($\star$5) is trivial. For the right to left direction, we merely tell how to instantiate the variables in V to make the left hand side true. For $0 \leq i \leq s$, define terms t_i, by induction on i, as follows: $t_0 = t$; $t_{i+1} = \lceil t_i + 1 \rceil$. Then we will interpret w_i by t_i. That this works is left to the reader to verify.

But granted this, we have shown how to replace the current main formula, by a quantifier-free formula, so if Case 3 ever arrives, we will be done. But Cases 1 and 2 can hold only finitely often since each time we pass through one of them the number of variables in V drops. So we will eventually succeed in eliminating all the quantifiers from our main formula. The treatment of Step 10, and with that the proofs of Lemma 3 and Theorem 2 are complete. $\boxtimes$

Once the quantifiers have been eliminated, we need to be able to decide the quantifier-free matrix. Thus, we have the following theorem.

THEOREM 5. There is an effective procedure that, given a quantifier free L^{e}-sentence q, produces a $k \geq 1$ and a truth value b such that for all k-good sets S, we have:

$$\mathbf{Z}_S \models q \leftrightarrow b.$$

Proof:

Let $S = \{s_0 < s_1 < s_2 < \cdots\}$. We first express each term t of q as a linear combination of $1, s_0, s_1, \ldots$, imposing requirements on k as we proceed to insure the correctness of these representations. Note that since q is a quantifier-free sentence, we do not have to worry about variables occurring in q.

We proceed by induction on the structure of t. The only non-trivial cases are where t is of the form $\lfloor t' \rfloor$ or $\lceil t' \rceil$. By the induction hypothesis, suppose that $t' = a + \sum_i a_i s_i$ (finite sum), where a and the a_i's are numeric constants. We impose the requirement that $k > |a| + \sum_i |a_i|$. We have the following cases:

1) All the a_i's are zero. In this case, $\lfloor t' \rfloor = 0$ and $\lceil t' \rceil = s_0$.

   For cases 2 through 6, let n be the largest index such that a_n is non-zero.

2) We have $a_n < 0$. In this case, $\lfloor t' \rfloor = 0$ (recall that the floor of a negative number is 0 by our definitions), and $\lceil t' \rceil = s_0$.

3) We have $a_n = 1$, but $a_{n-1} = \cdots = a_1 = 0$. If $a < 0$, then $\lfloor t' \rfloor = s_{n-1}$ (unless $n = 0$, in which case $\lfloor t' \rfloor = 0$), and $\lceil t' \rceil = s_n$. If $a = 0$, then $\lfloor t' \rfloor = \lceil t' \rceil = s_n$. If $a > 0$, then $\lfloor t' \rfloor = s_n$ and $\lceil t' \rceil = s_{n+1}$.

For cases 4 through 6, let $m < n$ be the next largest index such that a_m is non-zero.

4) We have $a_n = 1$ and $a_m < 0$. In this case, $\lfloor t' \rfloor = s_{n-1}$, and $\lceil t' \rceil = s_n$.

5) We have $a_n = 1$ and $a_m > 0$. In this case, $\lfloor t' \rfloor = s_n$, and $\lceil t' \rceil = s_{n+1}$.

6) We have $a_n > 1$. In this case, $\lfloor t' \rfloor = s_n$, and $\lceil t' \rceil = s_{n+1}$.

Once we have expressed all the terms of q in this fashion, we can determine the truth-value of the predicates using these terms by imposing further requirements on k. We have the following cases for atomic formulae:

1) $\mathcal{A}(t)$, where $t = a + \sum_i a_i s_i$. We impose the requirement that $k > |a| + \sum_i |a_i|$. This formula will be true iff the following three clauses are true: $a = 0$, there is exactly one non-zero a_i, and the value of that a_i is 1.

2) $D_j(t)$, where $t = a + \sum_i a_i s_i$. We impose the requirement that $j \leq k$. This formula will be true iff j divides a.

3) $t_1 = t_2$. We rewrite this as $t_1 - t_2 = 0$, where $t_1 - t_2 = a + \sum_i a_i s_i$. We impose the requirement that $k > |a| + \sum_i |a_i|$. This formula will be true iff $a = 0$ and all the a_i's are zero.

4) $t_1 > t_2$. We rewrite this as $t_1 - t_2 > 0$, where $t_1 - t_2 = a + \sum_i a_i s_i$. We impose the requirement that $k > |a| + \sum_i |a_i|$. This formula will be true iff the non-zero a_i with the largest index is positive, or no such a_i exists and $a > 0$.

Once all atomic formulae have been decided, it is trivial to decide the entire sentence.

$\boxtimes$

By combining Theorem 2 and Theorem 5 we obtain the main decidability result.

THEOREM 6. There is an effective procedure that, given an L^e-sentence q, produces a $k \geq 1$ and a truth value b such that all k-good sets S satisfy $\mathbf{Z}_S \models q \leftrightarrow b$.

Proof:

Use Theorem 2 on the L^e-sentence q to obtain a quantifier free L^e-sentence q' and a number k'. Then use Theorem 5 on q' to obtain a truth value b and a number k''. Take

$k := \max(k', k'')$ and let the truth value desired be b. ⊠

COROLLARY 7. There is an effective procedure that, given an L-sentence q, produces a $k \geq 1$ and a truth value b such that all k-good sets S satisfy $\mathbf{Z}_S \models q \leftrightarrow b$.

COROLLARY 8. It is decidable, for an arbitrary L-sentence q, to select one of q or $\neg q$ such that there are an uncountable number of sets which make the selected sentence true.

Proof:

By Theorem 6 one can effectively find k and b such that for all k-good sets q has truth value b. If $b = TRUE$ then pick q, else pick $\neg q$. Since, for any k, there are an uncountable number of k-good sets, the sentence chosen will be true for an uncountable number of sets. ⊠

A similar result applies for queries with a function symbol $\mathcal{F}$ in place of the set symbol $\mathcal{A}$, since any set can be interpreted as a 0-1 valued function. Thus, if there are an uncountable number of 0-1 valued functions satisfying some sentence, there are certainly an uncountable number of functions that satisfy that sentence.

3) REC $\notin$ QEX$[+, <]$

We need to extend Theorem 6 slightly to apply it to query learning. As it turns out, once we have been diagonalizing against a QIM for a while, we will have an initial prefix of the set which must be maintained, even though no extension of it is a k-good set. Thus, we define the corresponding sets which are k-good except for an initial prefix.

Notation: $\{0,1\}^*$ is the set of all finite sequences of 0's and 1's. If $\sigma \in \{0,1\}^*$ then for $i \geq 0$, $\sigma(i)$ denotes the $(i+1)^{\text{st}}$ bit of σ, and $|\sigma|$ denotes the length of σ.

Definition: For $k \geq 1$ and $\sigma \in \{0,1\}^*$, a (k,σ)-good set is a set $S \subseteq \mathbb{N}$ satisfying the following two conditions:

$$\{x \in S : x \geq |\sigma|\} \text{ is } k\text{-good},$$

$$(\forall x < |\sigma|)[x \in S \Leftrightarrow \sigma(x) = 1].$$

We now define the operation of "splicing" a given prefix on a set.

Definition: For $S \subseteq \mathbb{N}$ and $\sigma \in \{0,1\}^*$,

$$\sigma S = \{x : \sigma(x) = 1\} \cup \{x \in S : x \geq |\sigma|\}.$$

For the record, we state the following obvious lemma.

LEMMA 9. Let $k \geq 1$, $S \subseteq \mathbb{N}$, $\sigma \in \{0,1\}^*$, and $S \cap \{0, 1, \ldots, |\sigma|-1\} = \emptyset$. Then S is k-good iff σS is (k, σ)-good.

As Theorem 2 and Theorem 5 were proved for k-good sets, and not (k, σ)-good sets, we must first eliminate the prefix σ.

THEOREM 10. There is an effective procedure that, given an L-formula q and a $\sigma \in \{0,1\}^*$, produces an L-formula q' such that for all infinite sets $S \subseteq \mathbb{N}$ and all variable assignments $\vec{v}$, we have:

$$\mathbf{Z}_{\sigma S} \models q[\vec{v}] \iff \mathbf{Z}_{\sigma S} \models q'[\vec{v}] \iff \mathbf{Z}_S \models q'[\vec{v}].$$

Proof:

In q, replace each sub-formula of the form $\mathcal{A}(t)$ with

$$((t \geq |\sigma|) \wedge \mathcal{A}(t)) \vee \bigvee_{\sigma(i)=1} (t = i).$$

Note that $|\sigma|$ and i are meta-symbols in the above formula, representing constant symbols of the appropriate values. $\boxtimes$

THEOREM 11. There is an effective procedure that, given an L-sentence q and a $\sigma \in \{0,1\}^*$, produces a $k \geq 1$ and a truth-value b, such that all (k, σ)-good sets S satisfy:

$$\mathbf{Z}_S \models q \leftrightarrow b.$$

Proof:

Use Corollary 7 and Theorem 10. $\boxtimes$

To apply this result in inductive inference, we need to make a slight modification. Recall that the queries made by a QIM have variables that range over the natural numbers, whereas the variables in the above result range over the integers. However, since we have the *less than* predicate in the language, we can map any QIM query to an equivalent L-formula by explicitly requiring that each variable be non-negative.

In the following theorem, recall that $RECSET$ is the concept class consisting of all recursive sets.

THEOREM 12. $RECSET \notin QEX[+,<]$.

Proof:

The proof mirrors that of $RECSET \notin QEX[S,<]$ (Theorem 23 of [7]), where the part of the ω-automata is taken over by the values of k and σ. These values will be such that any (k,σ)-good set will satisfy our responses to the queries issued by the QIM so far, and will thus restrict the rest of the set without pinning it down precisely.

Given a QIM M that asks questions in the language $[+,<]$, we construct a recursive set A such that either A is not inferred by M, or A is finite but another set B ends up being constructed that is not inferred by M. The construction is in finite stages of effective extension.

At any stage s, we will have the following objects:

- A value of $\sigma_s \in \{0,1\}^*$ which represents the prefix of A that has been determined by stage s.
- A value of k_s such that any (k_s,σ_s)-good set will satisfy the responses we have given to the QIM M for the first s queries.

CONSTRUCTION

Initially, we take $k_0 := 1$ and σ_0 empty.

At stage $s+1$, we perform the following three steps:

1) Simulate M to determine the $(s+1)^{\text{st}}$ query and guess.
2) Use Theorem 11 to find k and truth value b. Answer the query with b.

3) Set $k_{s+1} := \max(k_s \cdot |\sigma_s|, k)$ Finally, we must determine σ_{s+1}. First, we extend σ_s by adding a string of 0's followed by a 1, such that the resulting prefix is consistent with the (k_{s+1}, σ_s)-good requirement. We do this at every stage to insure that A is infinite.

Next, we add another string of 0's to skip to the next place where we can add a 1 and still be (k_{s+1}, σ_s)-good. Let x be this place. We seek to diagonalize against the current guess, while constructing a default function in case the current guess diverges at x.

Let g be the current guess of QIM M. Perform the following in parallel:

4a) *Diagonalize against current guess.* Run $\phi_g(x)$. If this computation terminates before a mind change is found in (4b), then extend the prefix to σ_{s+1} by adding $1 \dot{-} \phi_g(x)$.

4b) *Look for a mind change; construct a default set.* Without permanently making any changes to A, continue simulating the QIM M on larger and larger prefixes, using Theorem 11 to answer the queries and determine the appropriate values of k. In between queries add a string of 0's followed by a 1 such that the resulting prefix is consistent with the appropriate goodness constraint. We do this to ensure that, if this substage goes on forever and a default set is constructed, it will be infinite. If this process causes M to change its mind before (4a) terminates, then actually use the sequence of answers to M's queries and the extensions to A that were needed to insure the correctness of those answers to extend the set A, and adjust k appropriately. Otherwise, if (4a) terminates before we find a mind change, then none of the actions in this sub-stage take effect: the extensions to A are not made, and the answers to the M's queries are not given.

END OF CONSTRUCTION

If neither (4a) or (4b) terminates at some stage s, then the infinite computation in (4b) gives us a recursive set B for which the QIM's guess diverges on x. Hence, M does not infer B.

If either (4a) or (4b) terminates at all stages s, then we have two cases. If (4b) was encountered an infinite number of times, then the QIM does not converge to a guess on recursive set A, which is the limit of the σ_s prefixes. Otherwise, the QIM converges to a guess which is wrong infinitely often on A.

Note that in either case the set constructed has the necessary (σ, k)-good properties to insure that all the answers to queries that were given are valid. $\boxtimes$

The notions of anomalies, behaviorally correct inference [1], and teams [10], can be naturally combined with our notion of query inference. Once these notions are understood, it is easy to define the classes $[1,n]Q_iBC^a[L]$. Using the techniques of Theorem 12 with those of the above referenced papers it is possible to show that for all $n, a \in \mathbb{N}$, $REC \notin [1,n]QBC^a[+,<]$.

4) PRIMREC $\notin$ $\mathbf{QEX_a}[+,<]$

In this section, we do a "primitive recursive version" of the construction in Theorem 12, to obtain that for any a, $PRIMREC \notin \mathrm{QEX}_a[+,<]$. It is not hard to show (by a "delay till ready" argument) that we may assume that all QIM's compute primitive recursive functions. By analyzing the proofs given in Section 2 (and using the fact that the decision procedure for Presburger Arithmetic is primitive recursive) one can show that the effective procedure proven to exist in Theorem 11 is primitive recursive. However, it is not primitive recursive to run $\phi_g(x)$ (step $4a$ of the construction in Theorem 12). We get around this by being nonconstructive: we will construct 2^{a+1} sets, one of which will work. Intuitively, each set constructed will have preset guesses as to what to do where, in the construction of Theorem 12, a Turing machine would be simulated.

Let $PRIMRECSET$ be the concept class consisting of all primitive recursive sets.

THEOREM 13. For any $a \geq 0$, $PRIMRECSET \notin \mathrm{QEX}_a[+,<]$.

Proof:

Let M be a QIM that makes queries in the language $[+,<]$. We may assume that M computes a primitive recursive function.

For every $\tau \in \{0,1\}^{a+1}$, we construct a primitive recursive set A_τ. Recall that $\tau(i)$ is the $(i+1)^{\mathrm{st}}$ bit of τ.

CONSTRUCTION of A_τ

Initially, we take $k_0 := 1$, σ_0 empty, $NUMMC = -1$, ($NUMMC$ is Number of Mind Changes), and $g_0 = \perp$ (the first guess is $\perp$).

At stage $s+1$, we perform the following three steps:

1) Simulate M to determine the $(s+1)^{\text{st}}$ query and guess.
2) Use Theorem 11 to find k and truth value b. Answer the query with b.
3) Set $k_{s+1} := \max(k_s \cdot |\sigma_s|, k)$.

Finally, we must determine σ_{s+1}. First, we extend σ_s by adding a string of 0's followed by a 1, such that the resulting prefix is consistent with the (k_{s+1}, σ_s)-good requirement. We do this at every stage to insure that A is infinite.

Next, we add another string of 0's to skip to the next place where we can add a 1 and still be (k_{s+1}, σ_s)-good. Let x be this place.

Let g be the current guess of QIM M, and let g' be the guess in the previous stage. If $g = g'$ then go to the next stage. If $g \neq g'$ then $NUMMC := NUMMC + 1$. If $NUMMC > a$ then let A_τ be some primitive recursive set which is (k_{s+1}, σ_s)-good and stop the construction. If $NUMMC \leq a$ then set $\sigma_s(x) = \tau(NUMMC)$.

END OF CONSTRUCTION.

It is easy to see that A_τ is primitive recursive. We show that there exists τ such that either M does not infer A_τ or M changes its mind more than a times while trying to infer A_τ.

Note that for all τ the construction of A_τ is identical until M outputs a non-null guess. Let s_1 be the stage where M outputs its first non-null guess g_1, and let x_1 be the x in the construction at stage s_1. Let $b_1 \in \{0,1\}$ be such that $\phi_g(x_1) \neq b_1$. (This is nonconstructive.)

Inductively assume that for $j \leq a+1$, there exist stages $s_1, s_2, \ldots, s_j$, guesses $g_1, \ldots, g_j$, bits $b_1, b_2, \ldots, b_j$, and numbers $x_1, \ldots, x_j$, such that if τ begins with $b_1 \cdots b_j$, then in the construction of A_τ, for all i such that $1 \leq i \leq j$:

a) At stage s_i guess g_i is made. No other non-null guesses are made at any stage $s \leq s_j$.

b) At stage s_i, the number x used in the construction is x_i and $\phi_{g_i}(x_i) \neq b_i$.

If $j \leq a$ and there exists a stage $s_{j+1} \geq s_j$ such that $g_{j+1} \neq g_j$, then let x_j be the number x used in that stage of construction. Set $b_{j+1} \in \{0,1\}$ such that $\phi_{g_j}(x_{j+1}) \neq b_{j+1}$. (This is nonconstructive.) It is easy to see that this value of b_{j+1} satisfies the conditions above.

If $j = a+1$ and there exists a stage $s_{j+1} \geq s_j$ such that $g_{j+1} \neq g_j$, then while M is trying to infer $A_{b_1 \cdots b_j}$, M changes its mind more than a times.

If $j \leq a+1$ and no such s_j exists, then M fails to infer $A_{b_1 \cdots b_j}$ because M's guess is g_j which is wrong at x_j.

Hence, there exists τ such that M fails to infer A_τ with $\leq a$ mind changes. $\boxtimes$

5) $\mathbf{Q_1EX_0[L] \subset EX}$

We show that for any reasonable language L, and any $c \in \mathbb{N}$, $\mathrm{EX}_{c+1} - \mathrm{Q}_1\mathrm{EX}_c[L] \neq \emptyset$. The key intuition used in the proof is that an existential query can only ask about a finite number of function values. In particular, we will see that the query

$$(\exists x, k)[\mathcal{F}(x) = x+1 \wedge \mathcal{F}(x+1) = x+2 \wedge \cdots \wedge \mathcal{F}(x+k-1) = x+k \wedge \mathcal{F}(x+k) = x]$$

cannot be phrased as an existential query in any reasonable language.

Throughout this section, let L denote an arbitrary but fixed reasonable language. Any query mentioned is an existential query in the language L.

Convention: If θ is an existential query then, by introducing new variables, we can rewrite θ so that the only terms that appear as arguments to $\mathcal{F}$ are variables. For example,

$$(\exists x_1, x_2)[\mathcal{F}(x_1^{x_2} + \mathcal{F}(x_1)) = 2 \wedge \mathcal{F}(8) < x_2]$$

becomes

$$(\exists x_1, x_2, x_3, x_4, x_5)[x_3 = x_1^{x_2} \wedge x_4 = x_3 + \mathcal{F}(x_1) \wedge x_5 = 8 \wedge \mathcal{F}(x_4) = 2 \wedge \mathcal{F}(x_5) < x_2].$$

Hence, we assume that all queries are of the form $(\exists x_1, \ldots, x_m)[\psi(x_1, \ldots, x_m, \mathcal{F}(x_1), \ldots, \mathcal{F}(x_m))]$.

Definition: A finite function is a *cycle* if it is of the form

$$\{(a+i, a+i+1) \mid 0 \leq i < k\} \cup \{(a+k, a)\},$$

where $a, k \in \mathbb{N}$ and $k \geq 1$. The cycle above has *starting point* a, and *length* k. It is denoted by $C(a,k)$.

Definition: A partial function *contains a cycle* if it has a cycle as a subfunction. A function is *cycle free* if it does not contain a cycle.

Lemma 14. Let θ be an existential query, σ be a finite initial segment such that $\sigma(0)=0$, and a be the least number where σ is not defined. Assume that for all functions h such that h extends σ and $(h-\sigma)$ is cycle free, h does not make θ true. Then there exists $k \geq 1$ with the following property: if g extends σ and the only cycles in $(g-\sigma)$ are of length $\geq k$, then g does not make θ true.

Proof: Let $\theta = (\exists x_1, \ldots, x_m)[\psi(x_1, \ldots, x_m, \mathcal{F}(x_1), \ldots, \mathcal{F}(x_m))]$. We claim that $k=m+1$ will suffice. Assume not. Then there exists a function g that extends σ such that the only cycles in $(g-\sigma)$ are of length $\geq k$ and g makes θ true. Hence, there exist $d_1, \ldots, d_m$ such that $\psi(d_1, \ldots, d_m, g(d_1), \ldots, g(d_m))$ is true. We use these $d_1, \ldots, d_m$ to define a function h that violates the premise.

Let h be

$$h(x) = \begin{cases} \sigma(x) & \text{if } x \text{ is in domain}(\sigma); \\ g(d_i) & \text{if } x = d_i; \\ 0 & \text{otherwise.} \end{cases}$$

Note that h extends σ, and $(h-\sigma)$ is cycle-free ($\sigma(0)=0$ is used here). Since for all i, $g(d_i) = h(d_i)$, and $\psi(d_1, \ldots, d_m, g(d_1), \ldots, g(d_m))$ is true, $\psi(d_1, \ldots, d_m, h(d_1), \ldots, h(d_m))$ is true. Hence, h makes θ true, which violates the premise. $\boxtimes$

In the proof below we will be simulating a QIM M that asks existential queries. We will answer these queries as best we can, but could be wrong. The definition below will help us approximate the answers.

Definition: Let $\sigma \in \mathbb{N}^*$ let $\theta = (\exists x_1, \ldots, x_m)[\psi(x_1, \ldots, x_m, \mathcal{F}(x_1), \ldots, \mathcal{F}(x_m))]$. We say that σ *believes θ is true* if there exist $d_1, \ldots, d_m$ such that $\sigma(d_1), \ldots, \sigma(d_m)$ are all defined, and $\psi(d_1, \ldots, d_m, \sigma(d_1), \ldots, \sigma(d_m))$ is true. We say that σ *believes θ is false* if it is not the case that σ believes θ is true. Note that if σ believes θ is true, then for any function f that extends σ, f makes θ true, but an analogous statement for falsity does not hold. Also, note that since L is a reasonable language, one can, given (σ, θ), recursively determine whether σ believes θ is true or believes θ is false.

Theorem 15. Let L be a reasonable language and let $c \in \mathbb{N}$. Then $\mathrm{EX}_{c+1} - \mathrm{Q}_1\mathrm{EX}_c[L] \neq \emptyset$.

Proof: Let

$$\begin{aligned}C = &\{f \mid \phi_{f(2)} = f \wedge (f \text{ contains no cycles })\} \cup \\ &\{f \mid (\exists d)[\,(1 \leq d \leq c+1) \wedge (f \text{ contains exactly } d \text{ cycles}) \\ &\qquad \wedge (\text{if } C(a,k) \text{ is the cycle with the largest starting point, then } \phi_{f(a+k+1)} = f)]\}.\end{aligned}$$

We show that $C \in \mathrm{EX}_{c+1} - \mathrm{Q}_1\mathrm{EX}_c[L]$.

We show that $C \in \mathrm{EX}_{c+1}$ as follows. Assume $f \in C$. To infer f with $\leq c+1$ mind changes, initially output $f(2)$; if one or more cycles are found in an initial segment of f, then let $C(a,k)$ be the cycle with the largest starting point, and output $f(a+k+1)$. Since there are $\leq c+1$ cycles, there will be at most $c+1$ mind changes.

We show $C \notin \mathrm{Q}_1\mathrm{EX}_c[L]$. Let M be a QIM that asks existential queries in language L and changes its mind $\leq c$ times (by convention we assume that if it changes its mind c times, it stops asking questions). We construct a function $f \in C$ that M does not infer. The finite initial segment of f determined prior to stage s is denoted f^s. The least number not in the domain of f^s is denoted a^s. As M asks queries, we supply answers that may later turn out to be incorrect. The sequence of responses given prior to stage s is denoted $\vec{b}^s \in \{0,1\}^*$. A sequence $\vec{b}^s$ is called *correct* if it is the sequence of correct answers to the first $|\vec{b}^s|$ queries made by M while trying to infer f. We will be building, on the side, a set D^s of numbers, which intuitively have been disqualified before stage s from being indices for f. If we discover that the answer to the ith query is incorrect, then the vector of responses is shortened to length i and the simulation is continued from the $(i+1)^{\mathrm{st}}$ query. We need to do this since the queries asked from that point may be different from what we had previously thought. If $\vec{b}^s$ is correct, then for all $t \geq s$, the first $|\vec{b}^s|$ bits of $\vec{b}^t$ are the correct answers to the first $|\vec{b}^s|$ queries made by M while trying to infer f.

Our strategy for supplying an answer to an existential query is to assume that it is false until there is a stage s such that f^s believes it, at which point we supply the answer true. Since the queries to which we answer false may change status, it is important to keep track of them. The disjunction of all the queries we are claiming are false prior to stage s is denoted θ^s.

During nonzero even stages, we will check if the answers supplied still appear valid. This is the only time we perform checking. During odd stages, we attempt to either diagonalize against the most recent guess of M, or force M to change its mind. The construction may have some nonconstructive stages, but at most $c+1$ of them, so the resulting function is recursive. Stages that are not nonconstructive will be called constructive.

We will make at most $c+2$ attempts at constructing the recursive function f. Each of these attempts will be a recursive construction, and so may be assumed to know (via the recursion theorem) its own Gödel number. If, for example, the first construction bogs down (and so does not produce a total recursive function) we will start the second construction with a (carefully and non-constructively chosen) finite extension of the finite function with which the first construction terminated. In this case, the number that, by the recursion theorem, was thought to be an index for the function being constructed, ends up being an index for a finite function.

CONSTRUCTION

Stage 0: By implicit use of the recursion theorem, assume we know e_0, an index for the function being constructed. (If the construction has a nonconstructive stage, and s' is the least such stage, then e_0 will be an index for the finite initial segment $f^{s'}$. See the commentary immediately proceding the construction.)

Let

$$\begin{aligned} f^0 &:= \{(0,0),(1,1),(2,e_0)\}, \\ D^0 &:= \emptyset. \end{aligned}$$

(We map 0 to 0 and 1 to 1 so that if we map other numbers to 0 or 1, a cycle cannot be created.)

Stage s (s nonzero and even): Let $\vec{b}^s = b_1 b_2 \cdots b_m$ where $b_i \in \{0,1\}$. Let q_i be the query M makes on input $b_1 b_2 \cdots b_{i-1}$, so b_i is the alleged answer to q_i. If there exists an i, $1 \le i \le m$, such that $b_i = 0$ and f^s believes that q_i is true, then let i_0 be the least such i, and set

$$\vec{b}^{s+1} := b_1 b_2 \cdots b_{i_0 - 1} 1,$$

$$\theta^{s+1} := \bigvee_{i=1, b_i = 0}^{i_0} q_i.$$

If this does not occur, then $\vec{b}^{s+1} := \vec{b}^s$ and $\theta^{s+1} := \theta^s$. In either case, set $f^{s+1} := f^s$.

Stage s (s odd): Let e and q be the guess and conjecture that M makes on input $b_1 b_2 \cdots b_m$ (where $b_i \in \{0,1\}$ is interpreted as the answer to the i^{th} query). If $e \in D^s$ then let $\vec{b}^{s+1} := \vec{b}^s 0$ and $\theta^{s+1} := \theta^s \vee q$, $f^{s+1} := f^s \cup \{(a_s, 0)\}$, and go to stage $s+1$. (The assumption that q is false will be checked during the next even stage.) If $e \notin D^s$ then we try to either diagonalize and make $f(a^s) \neq \phi_e(a^s)$, or make the vector $\vec{b}^s$ incorrect. (If neither is possible then we will take a drastic nonconstructive step.)

We simultaneously execute the following substages.

Substage 1: (Try to diagonalize.) Compute $\phi_e(a^s)$. If this computation converges at or before substage 2 terminates, then let

$$f^{s+1} := f^s \cup \{(a_s, 1 \dot{-} \phi_e(a_s))\},$$

$$\vec{b}^{s+1} := \vec{b}^s 0,$$

$$\theta^{s+1} := \theta^s \vee q,$$

$$D^{s+1} := D^s \cup \{e\}.$$

Substage 2: (Try to invalidate $\vec{b}^s$.) Look for a cycle free finite function τ such that the domains of f^s and τ are disjoint, $f^s \cup \tau$ has domain an initial segment, and $f^s \cup \tau$ believes θ^s is true. If such a τ is found before substage 1 terminates, then let z be the least number where $f^s \cup \tau$ is not defined and let

$$f^{s+1} := f^s \cup \tau \cup \{(z, 0)\}.$$

(The proper resetting of answers to queries will take place at some later even stage. The pair $(z, 0)$ ensures that if stage $s+2$ also uses substage 2, a cycle is not created by accident.)

If neither substage 1 nor substage 2 ever terminates, then we proceed nonconstructively. Since substage 1 never terminates, $\phi_e(a_s)\uparrow$. Since substage 2 never terminates, the triple (f^s, θ^s, a_s) satisfies the premise of Lemma 14. Let k be an integer at least as large as the value that Lemma 14 guarantees. (Also make k larger than any k value used in previous nonconstructive steps). We will ensure that the cycles in $(f - f^s)$ will be of length $\geq k$, so the final f will make θ^s false. Hence, the sequence $\vec{b}^s$ is correct. So e is a real conjecture made by M while trying to infer f (as opposed to a conjecture based on false answers to queries.)

There are two cases. Informally, either e is the $(c+1)^{\text{st}}$ conjecture made, so we define a total f such that $f \neq \phi_e$ (and end the construction), or e is not the $(c+1)^{\text{st}}$ conjecture in which case we define f^{s+1} such that no extension of f^{s+1} is equal to ϕ_e, and f^{s+1} contains a cycle and a new index for the rest of the construction.

Case 1: This is the $(c+1)^{\text{st}}$ nonconstructive stage. (We will later see that in this case, e is $M's$ final conjecture, and no more queries are going to be made.) If ϕ_e is almost always 0, then set

$$f := f^s \cup C(a_s, k) \cup \{(a_s + k + 1, e'_{c+1})\} \cup \{(x, 1) \mid x \geq a_s + k + 2\},$$

where e'_{c+1} is the index of this f; else set

$$f := f^s \cup C(a_s, k) \cup \{(a_s + k + 1, e''_{c+1})\} \cup \{(x, 0) \mid x \geq a_s + k + 2\},$$

where e''_{c+1} is the index of this f. The indices e'_{c+1} and e''_{c+1} both exist via the recursion theorem.

Case 2: This is the i^{th} nonconstructive stage, where $i < c+1$. If $\phi_e(a_s + k + 2)\downarrow$ then let $d = 1 \dot{-} \phi_e(a_s + k + 2)$, else let $d = 0$. (Since $d \in \{0, 1\}$, mapping a number to d will not accidentally cause a cycle.) Let e_i be the index of the function to be constructed in subsequent stages. (If the construction has a nonconstructive stage past s, and s' is the least such, then e_i will be an index for the finite initial segment $f^{s'}$. See also our remarks immediately proceding the construction.) Let

$$\begin{aligned} f^{s+1} &:= f^s \cup C(a_s, k) \cup \{(a_s + k + 1, e_i), (a_s + k + 2, d)\}, \\ \vec{b}^{s+1} &:= \vec{b}^s 0, \\ \theta^{s+1} &:= \theta^s \vee q, \\ D^{s+1} &:= D^s \cup \{e\}, \end{aligned}$$

and go to stage $s + 1$.

We put e into D^{s+1}, so that there will never be another nonconstructive stage where e is the index of interest. Since $\vec{b}^s$ is correct for any function that extends f^{s+1}, all later $\vec{b}^t$ agree with $\vec{b}^s$ on the first $|\vec{b}^s|$ bits; hence, the next time a nonconstructive stage is entered, the index of interest will be a conjecture that appears later than e in the sequence of conjectures that M makes while trying to infer f. It will follow that the true computation of M on f will make a mind change before any further nonconstructive stage.

END OF CONSTRUCTION

The number of nonconstructive stages in the above construction is $\leq c + 1$. If there are $c + 1$ nonconstructive stages, then the function f is defined in the last of them. If there are less than $c + 1$ nonconstructive stages, then the function f is defined as the limit of all the initial segments f^s. In either case, f is easily seen to be recursive.

We show $f \in C$. By the nature of the construction (helped by $f(0) = 0, f(1) = 1$, and almost all elements mapping to 0 or 1), the only cycles in f are those that we insert during a nonconstructive stage. If no nonconstructive stages are executed, then $f(2)$ is an index for f and there are no cycles; hence, $f \in C$. If d nonconstructive stages are executed, then there are $d \leq c + 1$ cycles, and if $C(a, k)$ is the one with the largest starting point, then $f(a + k + 1)$ is an index for f; hence $f \in C$.

We show that M does not infer f.

If there are $c + 1$ nonconstructive stages, then the last one was entered with e being the final conjecture made by M while trying to infer f. There are two points involved in seeing this. First, whenever we enter a nonconstructive stage s all the guesses recorded in $\vec{b}^s$ are correct. Second, each time we enter a non-constructive stage after the first one, there has been a mind change of M between the preceding nonconstructive stage and the present one. So by the time we enter the $c+1^{st}$ nonconstructive stage, M has made c mind changes and will never change his mind again. Since M cannot change its mind, and we made sure $f \neq \phi_e$, M cannot infer f.

If there are $\leq c$ nonconstructive stages, then there are an infinite number of stages. *Claim:* There exists an infinite set $T = \{s_1 < s_2 < \cdots\}$ such that $|\vec{b}^{s_n}| = n$ and $\vec{b}^{s_n}$ is correct.

Proof: We assume $s_1, \ldots, s_n$ exist and show that s_{n+1} exists. Let q be the query that M makes on input $\vec{b}^{s_n}$, and let

$$q = (\exists x_1, \ldots, x_m)[\psi(x_1, \ldots, x_m, \mathcal{F}(x_1), \ldots, \mathcal{F}(x_m))].$$

By the nature of s_n, for all $s \geq s_n$, the $(n+1)^{\text{st}}$ bit of $\vec{b}^s$ will be concerned with q.

If q is true of f, then there exists an m-tuple of numbers that satisfy ψ. Let $d_1, \ldots, d_m$ be the least such m-tuple (for definiteness) such that $\psi(d_1, \ldots, d_m, f(d_1), \ldots, f(d_m))$ is true. Let t be the least number such that for all i, $f^t(d_i)$ is defined. During the least even stage $s_{n+1} \geq \max\{t, s_n\}$, the existence of $d_1, \ldots, d_m$ will be noticed, and $\vec{b}^{s_{n+1}}$ will be set to $\vec{b}^{s_n}1$ which is correct.

If q is false of f then let s_{n+1} be the least odd stage greater than s_n. This will suffice, since the $(n+1)^{\text{st}}$ bit of $\vec{b}^{s_{n+1}}$ will be correctly set to 0, and never have any reason to change. $\boxtimes$(end of proof of claim.)

Let A be the set of constructive stages in T. It is easy to see that A is infinite. For all $s \in A$,

1) e^s is the conjecture made by M on input $\vec{b}^s$, and hence is a conjecture that M will make about f;
2) $\vec{b}^s$ is correct, so during stage s substage 2 will not be executed;
3) stage s is not nonconstructive.

Hence, either substage 1 is executed, which forces $\phi_{e^s} \neq f$, or it is observed that $e \in D^s$ which implies that it is known that $\phi_{e_s} \neq f$ from a previous stage. In either case, e^s is not an index for f. Hence, in the attempted inference of f by M, an infinite number of incorrect conjectures are made. Therefore, M does not infer f. $\boxtimes$

Note: The above proof also shows that $Q_2EX_0[S,<] - Q_1EX_0[L] \neq \emptyset$ since the set $C \in Q_2EX_0[S,<]$. The above proof can be modified to show that $\mathrm{EX}_{c+1} - [1, c+1]\mathrm{Q}_1\mathrm{EX}_0[L] \neq \emptyset$, where $[1, c+1]$ means $c+1$ teams of that type of machine (see [10] for first definition of team inference, [9] for the $[1,n]$ notation, and [6] for its use with query inference).

Corollary 16. $\mathrm{Q}_1\mathrm{EX}_0[L] \subset \mathrm{EX}$.

Proof: In [7] it was shown that for any reasonable language L, $\mathrm{Q}_1\mathrm{EX}_0[L] \subseteq \mathrm{EX}$. By Theorem 15, with $c = 0$, we obtain a proper inclusion. $\boxtimes$

Corollary 17. $\mathrm{Q}_1\mathrm{EX}_0[+,\times] \subset \mathrm{EX}$.

Theorem 18. $\mathrm{EX} - \bigcup_{c=0}^{\infty} \mathrm{Q}_1\mathrm{EX}_c[L] \neq \emptyset$.

Proof: Let

$$\begin{aligned} C = &\{f \mid \phi_{f(2)} = f \wedge (f \text{ contains no cycles })\} \cup \\ &\{f \mid (\exists d)[(1 \leq d) \wedge (f \text{ contains } d \text{ cycles }) \\ &\quad \wedge \text{ (if } C(a,k) \text{ is the one with the largest starting point, then } \phi_{f(a+k+1)} = f)]\}. \end{aligned}$$

The proof is similar to that of Theorem 15. $\boxtimes$

6) Open questions

In this paper, we have shown that for queries q (with one free set variable and no other free variables) in the language $[+,<]$, the problem of choosing one of q or $\neg q$ such that there is an uncountable number of sets that make the chosen query true, is decidable. The same question can be raised for other languages. In particular, we would like to know what happens if $L = [+,<,P2]$, where $P2$ is the predicate that tests if a term is a power of 2.

Although, it is known that $EX = \mathrm{Q}_0\mathrm{EX}[+,<] \subset \mathrm{Q}_1\mathrm{EX}[+,<] \subset \mathrm{Q}_2\mathrm{EX}[+,<]$ (the first two relationships were established in [7] and the last one in [6]), it is not known if this hierarchy extends any further. One candidate for a set in $\mathrm{Q}_3\mathrm{EX}[+,<] - \mathrm{Q}_2\mathrm{EX}[+,<]$ is $\{f \mid \phi_{f(0)} = f \wedge (f \text{ has a finite number of cycles})\} \cup$

$\{f \mid (f \text{ has an infinite number of cycles}) \wedge$

(the least cycle length that occurs infinitely often is an index for f).}

7) Acknowledgments

We would like to thank John Guthrie and Kathleen Romanik for proofreading.

REFERENCES

1. Case, J. and Smith, C.H. Comparison of Identification Criteria for Machine Inductive Inference. *Theoretical Computer Science 25* (1983), 193-220.
2. Daley, R.P. and Smith, C.H. On the Complexity of Inductive Inference. *Information and Control 69* (1986), 12-40.
3. Enderton, H.B. *A mathematical Introduction to Logic.* Academic Press, New York, 1972.
4. Ershov, Y.L., Lavrov, I.A., Taimanov, A.D., and Taitslin, M.A. Elementary theories. *Russian Math. Surveys 20* (1965), 35-105.
5. Fulk, M.A. Saving the Phenomena: Requirements that Inductive Inference Machines Not Contradict Known Data. *Information and Computation 79* (1988), 193-209.
6. Gasarch, W.I., Kinber, E., Pleszkoch, M. G., Smith, C.H., and Zeugmann, T. *Learning via Queries, Teams, and Anomalies.* Submitted to *Machine Learning* in 1990. Shorter version appeared in Third Annual Conference on Computational Learning Theory 1990..
7. Gasarch, W.I. and Smith, C.H. Learning via queries. *Journal of the Association of Computing Machinery* (To appear). Shorter version appeared in the Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science, 130-137, 1988..
8. Gold, E.M. Language Identification in the Limit. *Information and Control 10* (1967), 447-474.
9. Pitt, L. and Smith, C.H. Probability and Plurality for Aggregations of Learning Machines. *Information and Computation 77* (1988), 77-92.

10. Smith, C.H. The Power of Pluralism for Automatic Program Synthesis. *Journal of the Association for Computing Machinery 29, No. 4* (October 1982), 1144-1165.