JOURNAL OF COMPUTER AND SYSTEM SCIENCES 54, 487–501 (1997)
ARTICLE NO. SS971491

A Model of Interactive Teaching

H. David Mathias*, †

Department of Computer Science, Indiana Univeristy, Bloomington, Indiana 47405

Received June 6, 1995; revised October 18, 1996

Previous teaching models in the learning theory community have been batch models. That is, in these models the teacher has generated a single set of helpful examples to present to the learner. In this paper we present an interactive model in which the learner has the ability to ask queries as in the query learning model of Angluin. We show that this model is at least as powerful as previous teaching models. We also show that anything learnable with queries, even by a randomized learner, is teachable in our model. In all previous teaching models, all classes shown to be teachable are known to be efficiently learnable. An important concept class that is not known to be learnable is DNF formulas. We demonstrate the power of our approach by providing a deterministic teacher and learner for the class of DNF formulas. The learner makes only equivalence queries and all hypotheses are also DNF formulas.

1. INTRODUCTION

Most learning theory research is concerned, understandably, with worst-case analyses. That is, it is assumed that learning algorithms must interact with a hostile environment controlled by an omniscient adversary. While this is clearly very valuable it does not accurately model all learning environments. In particular, such modeling is not useful for determining how quickly learning problems may be completed. Models in which the environment is a helpful teacher are well suited to this task.

The interaction between learning algorithms and cooperative environments has been the subject of some research in the learning theory community. Much of this research has been directed at trying to develop models of teaching that are both useful and satisfying. For a teaching model to be useful it must be the case that it accurately reflects the relationship that can exist between learning algorithms in some applications and the "teachers" with which they interact. While the criteria for determining if a teaching model is satisfying are not universal, it should be the case that learners perform at least as well with helpful teachers as with adversarial ones; that a wide range of concept classes are teachable; and that the communication between the teacher and the learner is representative of that in domains in which such a model is likely to be used. We consider this last point in further detail.

When a cooperative teacher is introduced we want to ensure that the teacher is not avoiding "true learning" by, in some way, telling the answer to the learner. This problem is compounded when the model is made interactive since communication between the teacher and learner is bi-directional. Thus, we would also like to ensure that the learner is not similarly bypassing the learning task by, in some way, encoding "extra" information for the teacher. Clearly such *collusion* trivializes the learning task. The following examples illustrate why it is desirable to prevent collusion in models of teaching. First, consider the training of intelligent robot controllers. The task of directly programming a general purpose robot to perform specific chores may be extremely difficult for two reasons: (1) the operator thinks in Cartesian space while the robot's motions are rotations and joint movements, and (2) the robot's effector's are prone to calibration errors. For this application it would be helpful to have a teacher/learner pair (T/L pair) in which the teacher (human operator) establishes for the robot a set of representative actions in order to improve the robot's rate of learning. Notice that the teacher cannot simply provide the learner with the answer in this case due to their different frames of reference (i.e., it is too difficult to program directly). For the same reason, the learner cannot easily give information to the teacher other than its current estimate of the task it is being trained to perform. Another example comes from the field of human–computer interaction where research is being done on modifying user interfaces using *programming by example*. In this case the teacher may be a nontechnical end-user who cannot program the interface (although direct programming is possible). Once again, it is necessary to model a situation in which direct programming (collusion) cannot occur. If direct programming is possible then teaching/learning is unnecessary. Our goal is to develop a methodology that is useful in situations in which direct programming is excessively labor intensive or infeasible. Thus, our desire to address collusion is not motivated by purely theoretical reasons.

* This research was performed while the author was a student at Washington University. Support was provided by NSF grant CCR-9357707 with matching funds from Xerox PARC and WUTA.

† E-mail: dmath@cs.indiana.edu.

0022-0000/97 $25.00

It is important to understand the distinction between *collusion* and *cooperation*. For the reasons outlined above, we want to disallow collusion between the teacher and learner. However, we are modelling a teacher/learner relationship in which the teacher is attempting to help the learner accomplish some task in as efficient a manner as possible. Thus, it is reasonable to expect some level of cooperation between the teacher and learner. As an example, such cooperation could take the following form (as it does in our DNF algorithm): the teacher and learner agree on the semantics of a good counterexample (one that relates a large amount of information); the teacher then agrees to provide such a counterexample as long as one exists. All such communication must occur before the choice of a target concept but is likely to be specific to the concept class being learned. Such cooperation is at the heart of helpful teaching.

In nearly all previous teaching models, in an effort to prevent collusion, the model was reduced to that of teaching a consistent learner (a consistent learner is one whose hypothesis is consistent with all counterexamples seen). In other words, the teacher must give evidence ruling out every possible concept other than the target concept. This, however, eliminates much of the intended advantage of using a helpful teacher since the teacher is being required to teach to the "lowest common denominator." In effect, the adversarial teacher has been replaced by an adversarial learner. In fact, if a teacher is required to teach any consistent learner then there are classes for which efficient learning algorithms are known but an exponential number of examples are required for teaching. Obviously, this is counterintuitive.

Jackson and Tomkins [18] introduced the notion of a T/L pair in which the teacher and learner were designed to cooperate with each other. The motivation for T/L pairs is clear. First, they capture the intuition of a one-on-one teacher–student relationship. A model using T/L pairs also provides a tool for establishing lower bounds on the number of examples required by a learning algorithm. Finally, as illustrated above, there are environments in which the learner and teacher speak different languages and, thus, encoding or programming are impossible and learning must be used. Goldman and Mathias [11] (GM) extended T/L pairs so that prevention of collusion does not reduce the model to teaching any consistent learner. In that model, the teacher prepares a teaching set—a collection of labeled examples that allow the learner to infer the target concept. To prevent collusion, an adversary is allowed to add to the teaching set any properly labeled examples. In this work we extend the idea of a T/L pair in the following, fundamental way. Communication between the teacher and learner is made interactive. It may not be obvious that this change enhances the teaching model. Consider that in an off-line teaching model the teacher must present all of the helpful examples at the beginning of the learning task. The collusion prevention scheme of Goldman and Mathias allows an adversary to add properly labeled examples to the teaching set prepared by the teacher. Consider the possible computations of the learner as a tree. In some cases, using one of the adversarial examples could place the learner in a subtree containing no successful terminations. If, however, the learner could ask a query, a helpful teacher could provide an answer (or answers to a series of queries) that would allow the learner to relocate itself to a substree containing successful computations. We return to this point in Section 5, when we show the power of such interaction using the work of Bshouty [6].

When we discuss the desirability of preventing collusion we beg the question of what constitutes collusion. Collusion is difficult to define formally. Unfortunately, in an interactive model of teaching, it is even more difficult to prevent. We devised several collusion prevention schemes (each more complex than the preceding one) but each was defeated by increasingly complex methods employed by the teacher and learner. We discuss one such method in the Appendix. While we cannot claim that prevention of collusion in such a model is impossible, it certainly appears to be quite difficult. In fact, we conjecture that there is no collusion prevention scheme that does not reduce the model to teaching a consistent learner (page 11). As we have seen, requiring a teacher to teach any consistent learner is counter to the goal of a teaching model. Therefore, in this work, we do not present a formal collusion prevention result. We provide an intuitive, and broad, statement of what constitutes collusion and show that the teacher/learner pairs we present do not engage in these types of communication. This seems a reasonable alternative to a theorem that no teacher/learner pair can collude since lack of such a result does not mitigate the intuitive appeal of *interactive* teaching. We discuss this further in Section 4.

The learnability of disjunctive normal form (DNF) formulas is among the biggest open problems in computational learning theory. DNF is an important class because it is rich in its representational power and because it is simple and natural. While many subclasses of DNF have been shown to be learnable not much progress has been made for the general case. Two recent results illustrate the state-of-the-art in DNF learning. Bshouty *et al.* [7] gave a *randomized* algorithm, using restricted subset and superset queries, to learn DNF. In the PAC model, Jackson [17] has given an algorithm using membership queries to learn DNF against the uniform distribution. We demonstrate the power of our approach by giving a deterministic teacher/learner pair for the class of DNF formulas. The learner runs in polynomial time and uses only equivalence queries. Furthermore, all hypotheses are from the class of DNF. Our algorithms are quite natural and rely on the simple, underlying structure of disjunctive normal form. We show that there does not exist

a teacher/learner pair for DNF in which both the teacher and learner run in polynomial time. Thus, any improvement in the complexity of our algorithms would be incremental. Note that while our model is powerful enough to allow the teaching of DNF formulas the model is quite reasonable—the learner asks questions that are answered by a helpful teacher.

The remainder of this paper is organized as follows. We first outline the relevant previous work. In Section 3 we give definitions that are useful in reading this paper. We then define our model in Section 4 and describe several general results. In Section 5 we discuss an interesting relationship between our model and the monotone theory of Bshouty. We then, in Section 6, give a deterministic teacher and learner for the class of disjunctive normal form formulas. The learner for this class runs in polynomial time and uses only equivalence queries. All hypotheses are DNF formulas. In this section we also discuss an extension of our algorithms to an interesting geometric class that generalizes DNF formulas. In Section 7 we explore several variations of the model. Finally, we conclude and list open problems.

2. PREVIOUS WORK

Goldman, Rivest, and Schapire [13] first introduced the model of teacher-directed learning, a variant of the on-line learning model, in which examples are chosen by a helpful teacher. Recently, Rivest and Yin [22] have demonstrated the power of a helpful teacher by giving concept classes that are efficiently teachable in the teacher-directed model but that are not efficiently learnable in the self-directed learning model of Goldman and Sloan [14]. Since the introduction of the teacher-directed model, several interesting models have been proposed to study the complexity of teaching. The first *formal* model of teaching was introduced by Goldman and Kearns [12]. In this model they defined the teaching complexity as the minimum number of examples that a teacher must present to *any* consistent learner to enable the learner to exactly identify the target concept. In independent work, Shinohara and Miyano [26] introduced a model in which a class is *teachable by examples* if there exists a sample of polynomial size that allows all consistent learners to achieve exact identification of the target concept.

The notion that, to avoid collusion, a teacher should be required to teach any consistent learner runs counter to the intuition motivating models of teaching. To remedy this, Jackson and Tomkins [18] introduced teacher/learner pairs. In their model a teacher and learner are paired together to cooperate. To prevent collusion they require that the learner must output a hypothesis logically equivalent to the target, or no concept at all, even if the teacher is replaced by an adversarial substitute. Unfortunately, Jackson and Tomkins showed that, due to this method of collusion prevention, the teacher must still teach any consistent learner. Their model also allows for the teacher and learner to share a small amount of information. These *trusted bits* allow the teacher to communicate a stopping condition or a size parameter of the target. When trusted bits are allowed, they show that any class that is learnable is teachable in their model.

Our model is derived from that of Goldman and Mathias [11]. They developed a model that pairs teachers and learners but prevents collusion without forcing the teacher to teach any consistent learner. In their model the teacher constructs a *teaching set* designed to teach optimally, to a particular learner, the target concept. To prevent collusion, an adversary is permitted to add to this teaching set properly labeled examples. They give one formal definition of collusion and prove that this adversary is able to prevent it. They also prove that anything that is deterministically learnable from example-based queries is teachable in their model. Theirs is the first formal model of teaching for which this is true without relying on additional information.

Aside from the work on formal models of teaching there has also been interest in complexity measures of various concept classes in existing learning models. Perhaps most general is the work of Hegedűs [15, 16] who defines several general combinatorial measures on the complexity of teaching. Anthony *et al.* [5] consider subclasses of linearly separable boolean functions. They compute bounds on the size of the smallest sample with which only the target function is consistent. Natarajan [21] defines a dimension measure for classes of Boolean functions that measures the complexity of a class $\mathscr{C}$ by the length of the shortest example sequence for which the target function is the unique, most specific function from $\mathscr{C}$ consistent with the sample. In a model by Salzberg *et al.* [25] a helpful teacher presents a shortest example sequence allowing the learner, using the nearest-neighbor algorithm, to learn the target concept. Romanik and Smith [23, 24] propose a testing problem in which the goal is to construct, for a given target concept, a set of examples such that any concept that is consistent with this test set is "close" to the target in a probabilistic sense.

There has been some work on teaching in the inductive inference community. Though neither presents a complete model of teaching, both Freivalds, Kinber, and Wiehagen [9] and Lange and Wiehagen [20] have examined inference from "good examples" chosen by a helpful teacher. By presenting the learner with a superset of the teaching set prepared by the teacher, encoding is prevented in both of these models. Lange and Wiehagen [20] examine learning pattern languages and show that this can be achieved with good examples.

3. PRELIMINARIES

The teaching model that we present in this paper is based on the model of learning with queries developed by Angluin

[1]. In this model the learner's goal is to infer an unknown *target concept* f chosen from known *concept class* $\mathscr{C}$. More precisely, $\mathscr{C}$ is a *representation class*, a set of representations of functions. Throughout this paper, however, we use *representation class* and *concept class* interchangeably. Typically, $\mathscr{C}$ is parameterized by a size measure, n, so that $\mathscr{C} = \bigcup_{n \geqslant 1} \mathscr{C}_n$. In this paper we consider Boolean concept classes, thus n is the number of boolean variables which we denote $y_1, ..., y_n$. Each representation $f \in \mathscr{C}$ has a size, denoted $\|f\|$, which is the number of bits required to write f as a member of the representation class from which it was drawn. An *instance* is an assignment to the n boolean variables. The *instance space* is denoted $\mathscr{X} = \bigcup_{n \geqslant 1} \mathscr{X}_n$, where $\mathscr{X}_n = \{0, 1\}^n$. Let c be a concept. Then $c \subseteq \mathscr{X}_n$. The learner's hypothesis, h, is a polynomially evaluatable function $\{0, 1\}^n \to \{0, 1\}$. Let h represent the learner's hypothesis. A learning algorithm achieves *exact identification* of a concept class if for all instances $x \in \mathscr{X}$, $h(x) = f(x)$.

The teaching model of Goldman and Mathias [11] is important to this work. In that model, a *valid T/L pair* consists of a teacher T and a learner L such that for any $f \in \mathscr{C}$ the teaching set $T(f)$ produced by the teacher has the property that if L is provided with any teaching set $A(T(f)) \supseteq T(f)$, where all added examples are properly labeled according to f, then any representation f' output by the learner will be logically equivalent to f. Notice that this is a batch model—the teacher produces a single batch of examples for presentation to the learner.

In our model the learner has the ability to make queries to learn about the target concept. The query types in most common use in query learning algorithms are membership queries and equivalence queries. In a *membership query* the learner asks for the classification according to the target concept of an instance of its choosing. In an *equivalence query* the learner asks if its current hypothesis is logically equivalent to the target concept. If this is not the case then, as defined above, a counterexample is returned. A *positive counterexample* is an instance x_i such that $h(x_i) = 0$ and $f(x_i) = 1$. A *negative counterexample* is symmetric. A *restricted* equivalence query is one that is answered "yes" or "no" but no counterexample is provided. Two other query types that we discuss are superset and subset queries. A *superset query* is answered "yes" if $\forall x \in \mathscr{X}$ such that $f(x) = 1$, $h(x) = 1$. A positive counterexample is returned otherwise. A *subset query* is answered "yes" if $\forall x \in \mathscr{X}$ such that $h(x) = 1$, $f(x) = 1$. A negative counterexample is returned otherwise.

An *example-based query*, as defined by Goldman and Mathias, is any query of the form: "$\forall (x_1, ..., x_k) \in \mathscr{X}^k$, does $\varphi_f(x_1, ..., x_k) = 1$?," where k is constant and $\varphi_f(x_1, ..., x_k)$ is any polynomially evaluatable predicate with membership query access to target concept f. The answer to an example-based query is "yes" or a counterexample $(x_1, ..., x_k) \in \mathscr{X}^k$ (with their labels) for which $\varphi_f(x_1, ..., x_k) = 0$ and the labeled instances for which membership queries were made to evaluate the predicate. The instances on which membership queries were made serve as witnesses for the counterexample. The class of example-based queries includes all queries in common use in exact identification algorithms (*e.g.*, membership, equivalence, superset, subset, disjointness, exhaustiveness). For these queries $k = 1$ (each predicate operates on a single instance) and no membership queries are required for evaluation.

To illustrate the concept of an example-based query we provide two examples: one of a common query type—a subset query, and the second of a rather bizarre query. Intuitively, in a subset query the learner asks if its hypothesis is a subset of the target concept. In other words, is it the case that there are no negative counterexamples? As an example-based query a subset query has the following form. As noted above, $k = 1$. The predicate, φ, being evaluated is: for hypothesis h, $h(x) = 1 \Rightarrow f(x) = 1$. The answer is "yes" if $\varphi = 1$ and is a counterexample otherwise. Note that no membership queries are required to evaluate φ. In contrast, we define a *minimal traversal query*, MTQ(x, d), as follows. Given instance x and integer $d \leqslant n$, the φ being evaluated is: all assignments obtained from x, by flipping at most d bits, are negative. Once again, $k = 1$. The answer to such a query is either "yes" or a positive example x' differing from x in $k \leqslant d$ bits. Additionally, the learner is given the instances that differ from x in fewer than k bits to verify that the minimal number of bits were flipped in x'. Obviously, this query is very powerful and is included here for illustrative purposes only.

The class of *disjunctive normal form* formulas (DNF) is important and is duscussed in this paper. A DNF formula f is a disjunction of some number of terms: $f = t_1 \vee t_2 \vee \cdots \vee t_m$. We use m to denote the number of terms. Each term is a conjunction of literals: $t_i = l_{i_1} \wedge \cdots \wedge l_{i_s}$ where each l_{i_j} is a variable or its negation. A *non-redundant* DNF formula is a DNF in reduced form. That is, no terms can be removed from the formula without altering the logical meaning of the formula.

4. THE TEACHING MODEL

In this section we describe our interactive model of teaching. In addition, we discuss in detail the issue of preventing collusion. We also provide several general results related to the model.

4.1. *Defining the Model*

As in the model of Goldman and Mathias, the teacher and learner can cooperate, prior to the start of the teaching session, in devising a teaching strategy for known concept class $\mathscr{C}$. Also participating in the teaching session is an

omniscient adversary with unbounded computational power.

To begin a teaching session the adversary selects a target function $f \in \mathscr{C}$ and passes it to the teacher. Stage i of a teaching session (which begins with the ith query) proceeds as follows:

1. The learner poses an example-based query and passes it to the adversary.

2. The adversary (knowing $\mathscr{C}$, f, T, and L) passes to the teacher a set of well-formed queries containing the query asked by the learner.

3. The teacher answers each query in the query set received from the adversary and passes this answer set to the adversary.

4. Next the adversary passes to the learner an answer set containing the answer set generated by the teacher as well as other correctly labeled answers.

The teaching session ends when the learner outputs a representation from hypotheses class $\mathscr{H} \supseteq \mathscr{C}$. A teaching session is illustrated in Fig. 1.

We use the following format for notation: Q denotes a query and R denotes an answer (or response) to the query. We use superscript L, T, and A to represent the learner, teacher, and adversary, respectively. We use subscript f to denote the function $f \in \mathscr{C}$ being learned. Thus, we let Q_f^L be the sequence of example-based queries asked by the learner for target function f and $Q_f^L[i]$ be the ith query in the sequence. For query $Q_f^L[i]$ asked by the learner the teacher receives from the adversary a set of queries, $Q_f^A[i] \supseteq \{Q_f^L[i]\}$, as described in item 2 above. We use Q_f^A to denote the sequence of these adversarial query sets for the entire learning task. The teacher's answer set for query set $Q_f^A[i]$ is denoted $R_f^T[i]$ and the sequence of these sets is R_f^T. $R_f^A[i] \supseteq R_f^T[i]$ represents the answer set presented to the learner by the adversary for query $Q_f^L[i]$, as described in item 4 above. If $R_f^A[i]$ does not contain an answer for $Q_f^L[i]$ then $Q_f^L[i]$ has been asnwered "yes." Let R_f^A represent the sequence of adversarial answer sets for the entire learning task. Stage i of the computation begins with $Q_f^L[i]$ and continues until $Q_f^L[i+1]$ is asked. Stage 0 lasts until the first query is asked.

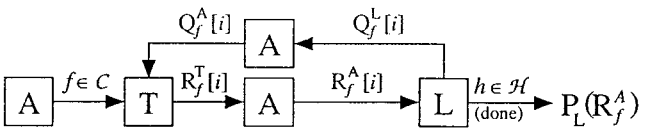

FIG. 1. An overview of a teaching session. When done, the learner outputs $h \in \mathscr{H} \supseteq \mathscr{C}$. For a randomized learner there are likely to be multiple $h \in \mathscr{H}$ with nonzero probability of being returned by the learner. However, all such hypotheses are logically equivalent to the target concept. (For a deterministic learner, $pr[h] = 1$ and $pr[h'] = 0$, $\forall h' \in \mathscr{H}$, $h' \neq h$.)

Let $|Y|$ denote the cardinality of set Y and $\|Y\|$ denote the number of bits needed to represent Y. Let s denote $|Q_f^L|$, the number of queries asked by the learner to learn $f \in \mathscr{C}$. In the following definitions we use notation similar to that of Goldman and Mathias. Let T be a teacher and L be a learner for concept class $\mathscr{C}$. As above, let R_f^T be the sequence of answer sets output by T and R_f^A be the sequence of answer sets produced by the adversary (where $R_f^A[i] \supseteq R_f^T[i]$, $\forall i$). Finally, let $P_L(R_f^A)$ be the probability distribution over $\mathscr{C}$ returned by L and let $\mathscr{H} \supseteq \mathscr{C}$. Then we say that T and L are a *valid* ${}_TI_L$ *pair* for $\mathscr{C}$ if for any $f \in \mathscr{C}$, any $f' \in \mathscr{H}$ with non-zero weight in $P_L(R_f^A)$ is logically equivalent to f. In other words, any representation output by L will be logically equivalent to f regardless of the actions of the adversary. We define T to be a polynomial-time teacher if for any query set $Q_f^A[i]$, presented to the teacher by the adversary and for any $f \in \mathscr{C}_n$, T outputs $R_f^T[i]$ in time polynomial in n, $\|f\|$ and $\|Q_f^A[i]\|$. If L asks a number of queries and runs in time polynomial in n, $\|f\|$ and $\max_{1 \leqslant i \leqslant s}\{\|R_f^A[i]\|\}$ then we say that L is a polynomial-time learner. We say that a representation class $\mathscr{C}$ is ${}_TI_L$*-teachable* if, for all $f \in \mathscr{C}_n$, there exists a valid ${}_{\mathrm{T}}\mathrm{I}_{\mathrm{L}}$ pair for which $|Q_f^L|$ (the number of queries asked by the learner) is polynomial in $\|f\|$, n and $\max_{1 \leqslant i \leqslant s}\{|R_f^A[i]|\}$ (the maximum number of bits required to represent any answer set provided by the adversary). If $\mathscr{C}$ is ${}_{\mathrm{T}}\mathrm{I}_{\mathrm{L}}$-teachable by a pair for which T is a polynomial-time teacher and L is a polynomial-time learner then we say that $\mathscr{C}$ is *polynomially* ${}_TI_L$*-teachable*. Finally, we say that $\mathscr{C}$ is *semi-poly* ${}_TI_L$*-teachable* if it is ${}_{\mathrm{T}}\mathrm{I}_{\mathrm{L}}$-teachable with a polynomial-time learner but a teacher that may be computationally unbounded.

Our model allows for randomized learners. This is feasible due to the interactive nature of the model—it is not necessary for the teacher to predict, *a priori*, all of the examples required by the learner. As with a deterministic learner the teacher knows the learner's algorithm and can provide a helpful answer. Unlike a deterministic learner, the output of a randomized learner defines a distribution on the hypothesis class that may have non-zero weight on multiple logically equivalent hypotheses. (Actually, a deterministic learner may output a randomized hypothesis in which case the output behavior is the same as that of a randomized learner.) Our model also allows for probabilistic teachers. The motivation for this is that it may be that we can devise a randomized polynomial time teacher for some classes for which there are no deterministic polynomial time teachers. A probabilistic teacher defines a distribution over the space of possible sequences of answers to the learner's queries.

It is easily seen that this model is robust against some types of noise. Specifically, the model can easily be extended to handle both incomplete membership queries [4] and malicious membership queries [3]. We briefly discuss the issue of noise in Section 7.

4.2. *Collusion*

As discussed in the Introduction, while we would like the teacher to help the learner accomplish the learning task as quickly as possible, we do not want the learning to take the form of encoding or some other form of collusion that clearly is not "real" learning/teaching. What should be meant by "collusion" or "real" learning is debatable. One motivation for models of teaching is that there may be instances that would reveal to the learner a great deal of information about the target concept. As an example, consider the concept class of a single, axis-parallel rectangle in the plane. A concept in this class may contain many positive instances. However, they are not all equally useful to a learner. In particular, a corner point of the target rectangle is likely to be of much more use to a learner than an arbitrary interior point. What we want to avoid, however, is the use of schemes in which the teacher can transmit information about a representation or encoding of the target concept, using instances that the learner can use without regard for their labels.

We distinguish two types of collusion: *answer collusion* and *query collusion*. Answer collusion occurs when the teacher passes "extra" information to the learner. Within answer collusion we define *intra-example* and *inter-example* collusion. Intra-example collusion occurs when the teacher is able to transmit to the learner, within a single example, information not pertaining to the logical function being taught. Inter-example collusion occurs when the teacher is able, using a sequence of examples, to transmit to the learner information not pertaining to the logical function. In inter-example collusion the learner is relying on some property of the examples presented, such as ordering. For intra-example collusion to occur it must be the case that the learner is relying on some property of the bits within an example such as ordering, parity or Hamming weight. In the boolean domain the teacher could pass up to n bits with a single example. This can be extended in nonobvious ways, using inter-example collusion, to pass longer sequences (we discuss this further in the next section). Since our model is interactive, we must also consider transmission of "extra" information from the learner to the teacher. We call this query collusion. Query collusion can be quite complex or as simple as the learner using query types to represent bits. In other words, by simply asking a query of a given type the learner could pass a bit to the teacher. As with answer collusion, query collusion can be divided into *intra-query* and *inter-query* collusion. We would like to include in the model a mechanism that is capable of preventing both answer and query collusion. Unfortunately, it may not be possible to devise such a mechanism for an interactive model (see Conjecture 1). We now give an intuitive definition of collusion.

STATEMENT 1. *A ${}_T I_L$ pair is said to collude if either of the following occurs*:

1. *The learner uses any example other than as a setting for the attributes of the domain or uses any example without regard to its classification.*

2. *The teacher uses any query other than as a request for an answer appropriate to that type of query.*

It is important to note that although our learner asks its queries on-line, in a fixed order, the ${}_{\mathrm{T}}\mathrm{I}_{\mathrm{L}}$ pair cannot rely on any order of the queries or the answers since the adversary can add any query or answer. For example, the adversary could add to the teacher's answer to the first query all of the answers that the teacher would give to all queries the learner would ask while learning f (recall that the adversary is omniscient). Clearly, this destroys the ability of the learner to use any ordering on the answers.

Although it may not be possible to achieve consensus about what constitutes collusion, we believe that the intuition for Statement 1 is clear. The examples provided by the teacher should be viewed as just that, instances (positive or negative) in the domain of some target function. We want to disallow any other use. Similarly, the queries asked by the learner should be seen by the teacher only as a request for information about the semantics of the target function. We also want to disallow any other use of queries. In the Appendix, we discuss one nonobvious method that can be employed by the teacher and learner to defeat some collusion prevention schemes.

Although we cannot currently prove that there does not exist a scheme that can prevent collusion in this model, we conjecture that this is the case.

Conjecture 1. There does not exist a collusion prevention scheme for this interactive teaching model that does not reduce the model to that of teaching a consistent learner.

4.3. *General Results*

We now present a series of results in this model that build on a variety of previous results. First, we show an interesting relationship to the GM model, namely that any class teachable in that model is teachable in this interactive model.

THEOREM 1. *Any representation class $\mathscr{C}$ that is T/L-teachable in the GM model is ${}_T I_L$-teachable in our interactive teaching model using only equivalence queries.*

Proof. We prove this theorem by construction. Let "GM learner" be the learner for some class $\mathscr{C}$ in the GM model. Let h_+ be a hypothesis that classifies as positive all positive examples seen and classifies all other instances as negative (this could take the form of a list of positive examples). h_- is defined symmetrically. Let h be the hypothesis of the GM learner. Note that the GM learner does not ask queries. Thus, h is its "final hypothesis" for the teaching set

is has been given. We construct our interactive learner from the following components: the GM learner, a memory that stores the answers to all queries asked (Mem), an algorithm for generating h_+ and h_- (minimal hyp) and a selector that chooses which hypothesis to use for the next equivalence query.

A learning session proceeds as follows: the learner begins by making an equivalence query with h, the initial hypothesis of the GM learner. A counterexample set is received, placed in the answer and passed to the generator for h_+/h_-. From this point on the learner alternates making equivalence queries with h, h_+, and h_-. An equivalence query asked with h_+ (respectively, h_-) allows the teacher to give a positive (respectively, negative) counterexample of its choice. Note that if the answer memory does not contain a valid teaching set then the behavior of the GM learner is unpredictable (it may not even halt). Thus, if the GM learner does not produce a hypothesis within time t (the required running time of the GM learner) then we do not use h for a query in that round. Since the only reason to use h is to detect completion, this does not affect the ability of our learner to learn.

At some point in this simulation the answer memory will contain all of the examples that a teacher would have placed in the teaching set for the GM learner. Note that our learner uses only equivalence queries. Clearly, the time used is polynomial if the time used by the GM learner is polynomial. See Fig. 2 for an illustration of this construction. ▌

Thus, we know that our interactive model is at least as powerful as the GM model. The following corollary is implied by Theorem 1 and by a theorem of Goldman and Mathias.

COROLLARY 1. *Any class that is polynomially learnable by a deterministic learner using example-based queries is semi-poly* ${}_T I_L$*-teachable using only equivalence queries.*

Ideally we would like to show separation of the two models by demonstrating a concept class, or family of concept classes, that is teachable in the interactive model

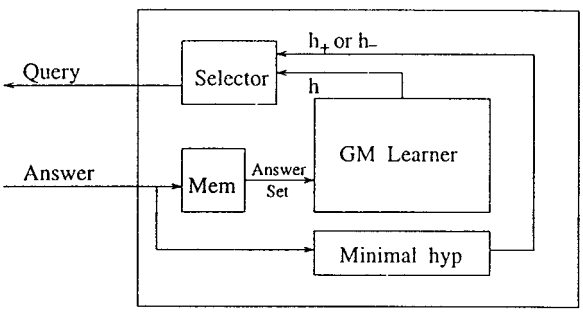

FIG. 2. The construction of the interactive learner that learns any class teachable in the GM model. See the proof of Theorem 1 for details.

but not in the GM model. While unable to do this, due to a lack of hardness results in that model, we do give evidence of separation by showing that any class learnable in polynomial time in the query learning model, even by a randomized learning algorithm, is teachable with a polynomial-time learner and a possibly computationally unbounded teacher in our interactive teaching model. Since Goldman and Mathias' proof that learnability implies teachability relies heavily on the ability of the teacher to simulate the learner, their theorem does not hold for randomized learners. Therefore, it is unclear if classes learnable by randomized learners are teachable in the GM model. Lending further evidence for separation is our ${}_T\mathrm{I}_L$ pair for DNF formulas. While this class has not been shown unteachable in the GM model it appears to be hard since much effort has failed to produce a positive result.

THEOREM 2. *Any representation class* $\mathscr{C}$ *probabilistically learnable in polynomial-time using example-based queries is semi-poly* ${}_T I_L$*-teachable using example-based queries.*

Proof. The proof is straightforward. Note that in the query learning model it is assumed that counterexamples are given by an omniscient adversary. Thus, by modifying the learning algorithm for $\mathscr{C}$ to handle counterexample sets, we have our ${}_T\mathrm{I}_L$ learner. Recall that a counterexample to an example-based query consists of a set of k instances. Since the adversary can add counterexamples, the learner must choose some k instances that comprise a valid counterexample. The learner can do this easily by simply evaluating the query (that is, $\varphi(x_1, ..., x_k)$), for all subsets of size k from the instances in the counterexample set, until a valid counterexample is found. ▌

Since in this model the teacher has full knowledge of the algorithm used by the learner, the teacher can simulate the learner assuming that the learner is deterministic. This allows the teacher and learner to simulate some types of queries with other types. For example, a superset query can be simulated using an equivalence query since the teacher knows that the learner needs a positive counterexample and will provide one. In fact, both superset and subset queries can be simulated this way since the teacher knows the type of counterexample the learner is seeking. If the learner is randomized, however, simulation is impossible since the teacher does not have knowledge of the random bits ("coin flips") used by the learner. In this case, we can simulate either subset *or* superset queries but not both (within one algorithm).

THEOREM 3. *Any representation class* $\mathscr{C}$ *(probabilistically) learnable in polynomial time using membership, equivalence, and subset or superset queries is semi-poly* ${}_T I_L$*-teachable using only membership and equivalence queries.*

Proof. Let $\mathscr{A}$ be a learning algorithm for $\mathscr{C}$. Our learner uses $\mathscr{A}$ replacing all subset or superset queries with equivalence queries. If $\mathscr{A}$ uses subset queries then the teacher answers each equivalence query with a negative counterexample if one exists. If the teacher returns a positive counterexample then the learner knows that the underlying subset query was answered "yes". Superset queries are symmetric. ▮

Thus, even if the learner is randomized, a ${}_{\mathrm{T}}\mathrm{I}_{\mathrm{L}}$ pair can simulate either subset of superset queries using equivalence queries. Some interesting results follow from the above theorems. The first of these uses a result of Bshouty, Cleve, Kannan, and Tamon [7] in which they show that DNF formulas and polynomial-size circuits are learnable by a randomized learner using only subset and superset queries. They do this by showing that equivalence queries and an NP oracle can be simulated using subset and superset queries. They show that the learner need not see counterexamples for the queries simulating the NP oracle (queries answered without counterexamples are known as *weak* queries).

THEOREM 4. *DNF formulas and polynomial size circuits are ${}_{T}I_{L}$-teachable with a randomized learner that uses only equivalence queries.*

Proof. The algorithm of Bshouty *et al.* uses subset and superset queries in pairs to simulate equivalence queries and an NP oracle. In our learning algorithm we replace the pair of queries simulating an equivalence query with a single equivalence query. We replace the pair of queries simulating the NP oracle with a pair of equivalence queries. The first query (simulating a superset query) is asked using the learner's hypothesis, h. The second query (simulating a subset query) is asked using $\bar{h}$. Since the random choices of the learner do not affect which type of query is asked, the teacher always knows whether the query it is answering is simulating a subset query or a superset query. Thus, the teacher can always answer these queries appropriately. ▮

In Section 6 we improve upon this result by demonstrating a more natural, deterministic ${}_{\mathrm{T}}\mathrm{I}_{\mathrm{L}}$ pair for the class of DNF formulas in which the learner uses only a polynomial number of equivalence queries *and* all hypotheses are from the class.

Another consequence of the results in this section follows from the work of Angluin [1] in which she gives an algorithm for learning pattern languages of length n. Her algorithm uses restricted superset queries and runs in time polynomial in n.

COROLLARY 2. *Pattern languages of length n are ${}_{T}I_{L}$-teachable in time polynomial in n with a learner that uses only equivalence queries.*

Proof. Once again the ${}_{\mathrm{T}}\mathrm{I}_{\mathrm{L}}$ pair can simulate the superset queries using equivalnece queries. Since the superset queries used by Angluin's algorithm are restricted, our learner can ignore the particular counterexample received and simply determine whether the counterexample is positive or negative. (For obvious reasons, it is not possible to simulate restricted superset queries using restricted equivalence queries.) ▮

The last result we discuss in this section also follows from a result of Angluin [1]. The "double sunflower" is a concept class defined by participants in the learning seminar at the University of California, Santa Cruz in the Fall of 1987. The class is defined as follows.

Let $N = 2^n$ for some given positive n. Let $X = \{x_1, ..., x_N\}$ and $Y = \{y_1, ..., y_N\}$. Let z_1 and z_2 be two instances not in X or Y. The instance space is $\mathscr{X} = X \cup Y \cup \{z_1, z_2\}$ and contains $2^{n+1} + 2$ instances. For each $j = 1, ..., N$ let concept $f_j = \{z_1, x_j\} \cup \{(Y - y_j)\}$. Thus, the hypothesis space is the set of all well-formed f_j. Note that only f_j contains x_j and does not contain y_j.

Angluin gives a proof that this concept class is not exactly identifiable by any learner with access to membership, equivalence, subset, superset, disjointness, and exhaustiveness queries using fewer than $N - 1$ queries.

COROLLARY 3. *The "double sunflower" is poly ${}_{T}I_{L}$-teachable with a learner that uses two equivalence queries.*

Proof. The learner uses any f_i from the hypothesis space as its initial hypothesis and makes an equivalence query. If f_i is the target concept then the query is answered "yes" and the learner is done. Otherwise, the teacher gives x_j as a positive counterexample telling the learner that f_j is the target. The next equivalence query is answered "yes." Note that this proof relies on the standard learning theory assumption that the learner knows the class being learned. ▮

The classes of Corollary 2 and Corollary 3 are also teachable in the GM model. These results are included here to illustrate the increased power of equivalence queries when answered in a helpful way.

5. RELATIONSHIP TO MONOTONE THEORY

One of the most interesting recent results in learning theory research is the development of the monotone theory, and its application to the learning of decision trees, by Bshouty [6]. Bshouty defines a complexity measure for concept classes called the *monotone dimension*, denoted M dim($\mathscr{C}$) for concept class $\mathscr{C}$. A set S of instances is a monotone basis for $f \in \mathscr{C}$ (denoted M-basis ($\{f\}$)) if f can be represented as a CNF formula such that every clause

in f is falsified by some instance in S. The $\text{M dim}(\mathscr{C}) = \max_{f \in \mathscr{C}}(\min(\text{M-basis}(\{f\})))$.[1] Thus, the maximum number of clauses in a minimal CNF representation of any $f \in \mathscr{C}$ is an upper bound on $\text{M dim}(\mathscr{C})$.

Bshouty then proves that, using $\text{size}_{\text{DNF}}(f)\ \text{M dim}(\mathscr{C})$ equivalence queries and n^2 membership queries for each equivalence query, any boolean function $f \in \mathscr{C}$ is learnable *if the monotone basis of the function is known to the learner*. He also gives a result in which the learner no longer needs to know the monotone basis. The running time of this algorithm is polynomial in $\text{size}_{\text{DNF}}(f)$, $\text{size}_{\text{CNF}}(f)$ and n. The hypothesis class used by these algorithms is dept-3 $\wedge$-$\vee$-$\wedge$ circuits. That is, each hypothesis H is the conjunction of a number of partial hypotheses H_i, each of which is a DNF formula.

In our model we can use the power of the teacher to allow the learner to use resources polynomial in the monotone dimension without knowing the monotone basis. Let N_A be the number of negative counterexamples added by the adversary. The following theorem states an important result implied by Bshouty's work and our model.

THEOREM 5. *Any $f \in \mathscr{C}$, for any concept class $\mathscr{C}$, is ${}_{T}I_{L}$-teachable using time and queries polynomial in* $\text{size}_{\text{DNF}}(f)$, $\text{M dim}(\mathscr{C})$, N_A *and n with a learner that does not know* $\text{M-basis}(\{f\})$.

Proof. Our learner is almost identical to Bshouty's learning algorithm with unknown monotone basis. Treating each negative counterexample as an element of the monotone basis, our learner simply creates a partial hypothesis (each partial hypothesis is a DNF) for each negative counterexample in the set. Thus, our learner creates at most $\text{M dim}(\mathscr{C}) + N_A$ partial hypotheses.

Refinements of the partial hypotheses are done by processing positive counterexamples. Bshouty showed that each partial hypothesis must be refined at most $\text{size}_{\text{DNF}}(f)$ times. Each refinement uses at most $O(n^2)$ membership queries. Thus, the number of equivalence queries used is $O((\text{M dim}(\mathscr{C}) + N_A)\ \text{size}_{\text{DNF}}(f))$ and the number of membership queries is $O((\text{M dim}(\mathscr{C}) + N_A)\ \text{size}_{\text{DNF}}(f)\ n^2)$. Finally, it is obvious that this ${}_{T}\text{I}_{\text{L}}$ pair does not collude since the learner was designed to work with an adversarial teacher and, therefore, makes no assumptions about the information it is receiving. ∎

Thus, if the adversary adds no negative counterexamples the learner uses time polynomial in $\text{M dim}(\mathscr{C})$ and $\text{size}_{\text{DNF}}(f)$. This is as opposed to time polynomial in $\text{size}_{\text{CNF}}(f)$ and $\text{size}_{\text{DNF}}(f)$ used by Bshouty's algorithm in the learning model. Thus, the improvement offered by the teaching model is significant since there are concept classes $\mathscr{C}$ such that there exist $f \in \mathscr{C}$ for which $\text{size}_{\text{CNF}}(f)$ is exponentially larger than $\text{M dim}(\mathscr{C})$. Unate DNF is such a class.

This result illustrates a key difference between this model of teaching and the GM model. As we discussed in Section 1, in some cases the use of an adversarial example in the GM model could irrecoverably sidetrack the learner. The monotone theory algorithm highlights this. In order to achieve equivalence with the target concept it is necessary that each partial hypothesis, H_i, in the learner's hypothesis be a superset of the target concept. That is, $\forall i,\ H_i \supseteq f$. In the GM model, however, it is possible that an adversarial negative instance is used to begin some H_j but that there are no positive examples in the teaching set that can be used to refine H_j. Thus, H_j will remain under-specified and never become a superset of f. Because the learner in the GM model does not have the ability to ask an equivalence query it can never delete or modify H_j and thus, the learner's hypothesis will never become logically equivalent to the target concept. We have been unable to create a T/L pair in the GM model using the monotone theory. Obviously, it is interactivity that gives this model its advantage.

In the next section we address the question of teachability of a class not known to be efficiently learnable. Specifically, we give a deterministic semi-poly ${}_{\text{T}}\text{I}_{\text{L}}$ pair for the class of DNF formulas. The learnability of this class is open.

6. TEACHING DNF FORMULAS

The learnability of disjunctive normal form formulas (DNF) is the subject of a great deal of learning theory research. The question remains one of the most important open questions in the field. The *teachability* of this class is either open or answered negatively in all previous models of teaching. DNF formulas have a very well-defined structure and it seems that a learning algorithm should be able to benefit from this. However, in typical learning models, the structure is shrouded by an omniscient adversary. In our teaching model a helpful teacher can select counterexamples that reveal the structure. In this section we present a semi-poly ${}_{\text{T}}\text{I}_{\text{L}}$ pair for DNF formulas, using only equivalence queries, where all hypotheses are DNF formulas.

It is helpful to consider the boolean instance space as a lattice. The top element of the lattice is the instance $\{1\}^n$ and the bottom element is the instance $\{0\}^n$. The elements are partially ordered by $\leqslant$, where $v \leqslant w$ if and only if each bit in v is less than or equal to the corresponding bit in w. The descendants (respectively, ancestors) of an instance v are all instances w such that $w \leqslant v$ (respectively, $w \geqslant v$).

Each DNF term, t_i, has a maximum (in the lattice) positive instance, $\max_i$, and a minimum positive instance, $\min_i$. Term t_i is specified by the "combination" of $\max_i$ and $\min_i$. That is, if $\max_i$ and $\min_i$ agree in a bit position then the corresponding literal is in t_i. For example, if

[1] Bshouty defines the monotone basis over the class rather than for each function in the class. However, the basis is always used as above, motivating this new definition.

$\max_i = 11101$ and $\min_i = 01100$ then $t_i = x_2 x_3 \overline{x_4}$. Given two instances v and w, we denote this operation by term(v, w).

Clearly, $\min_i$ and $\max_i$ are useful examples for building term t_i. Thus, our teacher provides to the learner, as counterexamples, $\min_i$ and $\max_i$ for each term of the target formula. If the learner knew the correct way to pair these counterexamples then it would have the target by simply combining as above. Notice, however, that the adversary can add counterexamples to the ones provided by the teacher. Therefore, the learner cannot rely on any ordering of the counterexamples it receives and does not know the correct pairing. Thus, the learner simply crosses the set of counterexamples on itself creating a term for each pair.

Operating in this way the learner will create a quadratic number of terms falling into three categories: *prime implicants*, *implicants*, *and nonimplicants*. All of the prime implicants and implicants can remain in the learner's hypothesis since they cause no counterexamples. Each of the nonimplicants must be deleted since each misclassifies at least one negative instance. The number of nonimplicants created is quadratic in the number of positive counterexamples seen by the learner. In the end, the learner's hypothesis is some DNF representation logically equivalent to the target formula.

Learner for DNF Formulas:

```
1.  h_c ← ∅
2.  h ← ∅
3.  Repeat
4.      V ← EQ(h_c)
5.      For each v ∈ V
6.          h_c ← h_c ∪ {v}
7.          For each t ∈ h_c
8.              h ← h ∨ term(v, t)
9.      V' ← EQ(h)
10.     If V' ≠ ∅
11.         For each v' ∈ V'
12.             If v' is positive
13.                 h_c ← h_c ∪ {v'}
14.                 For each t ∈ h_c
15.                     h ← h ∨ term(v', t)
16.             Else
17.                 Remove from h terms satisfied by v'
18. Until a V' contains no positive examples
19. While V ← EQ(h) ≠ ∅
20.     For each v ∈ V
21.         Remove from h all terms satisfied by v
22. Return(h)
```

FIG. 3. Our learner for DNF formulas. Hypothesis h_c is a minimal, consistent hypothesis that classifies all positive instances seen as positive and everything else as negative (this is equivalent to the disjunction of a singleton term for each positive instance seen). The hypothesis h is a DNF formula.

6.1. *The Teacher/Learner Pair*

We now present in detail our teaching and learning algorithms for the class of DNF formulas (Figs. 3 and 4).

Our learner uses multiple hypotheses, h_c and h; h_c is a consistent hypothesis that classifies as positive all positive instances seen and all other instances as negative (h_c can take the form of a DNF formula—a singleton term for each positive instance seen). Throughout a teaching session, h is a DNF formula approximating the target formula and is logically equivalent to the target formula at the end of the teaching session. An equivalence query made with h_c allows the teacher to gives as a counterexample the minimum or maximum positive instance for any term, provided that the learner has not already seen that instance.

The learner begins with $h = h_c = \varnothing$ and makes an equivalence query with h_c. The teacher always answers such a query with $\min_i$ or $\max_i$ for some term t_i. The learner will receive a counterexample set, including some number of adversarial counterexamples. Due to the nature of h_c each counterexample received is positive. For each counterexample v in the set the learner adds term(v, v) to h_c as a singleton term and then adds term(v, t) to h for each counterexample t in h_c. After all of the counterexamples in the set have been processed in this way an equivalence query is made with h. The teacher answers with a minimum or maximum positive instance, if one exists as a counterexample to the hypothesis. Note that it is possible that none are counterexamples to h since some already learned terms may contain these instances. If no such instance exists then the teacher answers with any positive counterexample. If there are no positive counterexamples then the teacher gives any negative counterexample. When this occurs, $h \supseteq f$ and the adversary can add only negative counterexamples also. Each negative counterexample, v, is used to delete from h those terms (nonimplicants) satisfied by v. After processing a counterexample set containing only negative counterexamples, the learner exits the "Repeat" loop and enters the "While" loop, continuing to ask equivalence queries with h. The learner receives negative counterexamples, until such a query is answered "yes" and the learner returns h and halts.

Teacher for DNF Formulas:

```
1.  Repeat
2.      For each EQ
3.          Answer min_i or max_i if a counterexample
4.          Else give any positive counterexample
5.          Else give any negative counterexample
6.          Else answer ∅
```

FIG. 4. Algorithm for the teacher for DNF formulas.

6.2. *Analysis*

In this section we prove the following theorem.

THEOREM 6. *There exists a deterministic* ${}_TI_L$ *pair that exactly identifies any target, f, in the class of* DNF *formulas using only equivalence queries. All hypotheses used by the learner are* DNF *formulas. The learner has query complexity polynomial in n, m, and* $|R_f^A|$ *and time complexity polynomial in n, m, and* $\|R_f^A\|$.

We first argue the correctness of our ${}_{\mathrm{T}}\mathrm{I}_{\mathrm{L}}$ pair using the following lemma.

LEMMA 1. *Our* ${}_TI_L$ *pair exactly identifies any* DNF *formula using only equivalence queries with hypotheses that are* DNF *formulas.*

Proof. We begin by examining the learner assuming that h_c contains $\min_i$ and $\max_i$ $\forall 1 \leqslant i \leqslant m$, as well as other positive instances. We then show that the learner can build h_c.

Each positive counterexample received by the learner is combined with every instance in h_c and the corresponding terms placed in h (the singleton term corresponding to the counterexample is placed in both h and h_c). Therefore, h contains a term for each pair of instances in a cross product of h_c with itself. Thus, $h \supseteq f$.

If equality holds then we are done. Otherwise, $h \supset f$. Since when presented with a negative counterexample the learner deletes any term satisfied and since any nonimplicant term is satisfied by at least one negative instance, all nonimplicant terms are deleted by negative counterexamples.

h_c classifies as negative any instance that the learner has not yet seen. This allows the teacher to provide as a counterexample to h_c any positive instance not already contained in h_c. Thus, after at most $2m$ equivalence queries with h_c, h_c will contain the minimum and maximum positive instances for each term of f. ∎

Thus, when our algorithm halts the learner's hypothesis is logically equivalent to the target formula. It may not be obvious that our teacher is unable to force the learner to output a particular DNF representation. Notice, however, that the adversary can add the minimum and maximum positive instances for terms in logically equivalent DNF representations. The resulting terms created by the learner are implicants of the target function and are, therefore, not deleted from the learner's hypothesis.

Next we argue that our ${}_{\mathrm{T}}\mathrm{I}_{\mathrm{L}}$ pair for this class is indeed a semi-poly ${}_{\mathrm{T}}\mathrm{I}_{\mathrm{L}}$ pair.

LEMMA 2. *Our deterministic* ${}_TI_L$ *pair for* DNF *formulas has query complexity polynomial in n, m, and* $|R_f^A|$. *The learner has time complexity polynomial in n, m, and* $\|R_f^A\|$.

Proof. We first examine the total number of terms added to h. Note that $|h_c| \leqslant |R_f^A|$. Since h contains exactly one (not necessarily unique) term for every tuple in the cross product of h_c with itself, the number of terms added to h is no more than $|R_f^A|^2$. Since every negative counterexample removes at least one nonimplicant from h, at most $(|R_f^A|^2 - m)$ negative counterexamples are required.

There are at most $2m$ minimum and maximum positive instances, therefore, the learner will iterate the Repeat loop at most $2m$ times. Within each iteration 2 equivalence queries are asked. Thus, the Repeat loop is responsible for $4m$ equivalence queries. Thus, the total number of queries asked by the learner is $O(|R_f^A|^2)$. Note that $|R_f^A| = \Omega(m)$.

Next, we bound the time used by our learner. Creating the terms to add to h takes $O(n)$ time each yielding a total time to create h of $O(n \cdot |R_f^A|^2)$. Processing the negative counterexamples requires $O(n \cdot |R_f^A|^2)$ time each $-O(n)$ time to check each of the $O(|R_f^A|^2)$ terms in h. Thus the total time to process all of the negative counterexamples is $O(\|R_f^A\|^4)$ which dominates the running time of the learner. ∎

The proof of Theorem 6 follows immediately from Lemma 1 and Lemma 2.

Finally, we claim that our teacher and learner for this class do not engage in collusion as it was described in Section 4. We provide intuition that neither answer collusion nor query collusion occur. The learner receives counterexamples at three places in the algorithm: lines 4, 9, and 19. In line 4, the counterexample is always positive and each is treated as a maximum or minimum positive instance for some term. In line 19, only negative counterexamples are received and each is used to delete all terms containing that counterexample. In line 9 the counterexamples seen may be either positive or negative but are processed, depending on sign, as above. Since no other processing is done on these examples, there is no answer collusion. To see that there is no query collusion notice that the teacher always gives a maximum or minimum positive instance if one exists as a counterexample. Otherwise, *any* positive counterexample or *any* negative counterexample is given. The teacher does not use the queries in any other way.

The time required by the teacher is not polynomially bounded. Next we show that no teacher for DNF in this model can run in polynomial time.

THEOREM 7. *There does not exist a* ${}_TI_L$ *pair for* DNF *with a teacher that runs in polynomial time unless* $\mathrm{P} = \mathrm{NP}$.

Proof. The proof is straightforward. Note that the teacher is required to answer arbitrary adversarial queries. By asking an equivalence query with the identically true hypothesis the adversary forces the teacher to determine if the target DNF is a tautology. This cannot be done in polynomial time unless $\mathrm{P} = \mathrm{NP}$. ∎

We note that in some sense this result subsumes the ${}_{\mathrm{T}}\mathrm{I}_{\mathrm{L}}$ pair using the monotone theory since any function can be taught in time polynomial in its DNF size without regard for its monotone dimension. However, for classes with small monotone dimension, that result may be more efficient than the algorithm in this section. DNF formulas are not such a class. In fact, even Read-Twice DNF have monotone dimensions that are exponential in the number of boolean variables in the domain.

6.3. *A Geometric Extension*

Unions of d-dimensional, axis-parallel boxes in discretized d-dimensional space generalize DNF formulas. We use the notation of Goldberg, Goldman, and Mathias [10] to define the class formally. $\textsc{box}_n^d$ denotes the class of axis-parallel boxes over $\{1, ..., n\}^d$. So d represents the number of dimensions and n represents the number of discrete values that exist in each dimension. Let $[i, j]$ denote the set $\{m \in \mathbf{N} \mid i \leqslant m \leqslant j\}$. Then, $\textsc{box}_n^d = \{ \times_{k=1}^d \mid 1 \leqslant i_k \leqslant j_k \leqslant n \}$. So i_k and j_k are the minimum and maximum positive values of the kth coordinate of a box. Note that by allowing equality of i_k and j_k we include in $\textsc{box}_n^d$ boxes with zero size in dimension k. Finally, let $\bigcup_s \textsc{box}_n^d$ denote the class of the union of at most s concepts from $\textsc{box}_n^d$. Given a box b, we define the corner c_l as the point on the boundary of b such that the kth coordinate of c_l is less than or equal to the kth coordinate of all other points in b for all $1 \leqslant k \leqslant d$. Conceptually, this is the point on b closest to the origin. The point, c_u, on b farthest from the origin is symmetrically defined. Structurally this class is quite similar to DNF. Each is a union (disjunction) of a number of substructures. Each of these substructures is easily specified by two instances—in the case of DNF formulas these are the minimum and maximum positive instances and in the geometric case these are c_l and c_u. Our ${}_{\mathrm{T}}\mathrm{I}_{\mathrm{L}}$ pair for DNF formulas is easily modified for this geometric class as we show in this corollary to Theorem 6.

COROLLARY 4. *There exists a deterministic ${}_T I_L$ pair that exactly identifies any target, f, in the class $\bigcup_s \textsc{box}_n^d$ using only equivalence queries where all hypotheses are unions of boxes. The learner runs in time polynomial in* $\lg n$, s, d, *and* $|R_f^A|$.

Note that if the boxes are not axis-parallel teaching is still possible, polynomial in $\lg n$, s, d, $|R_f^A|$, and the number of slopes, provided that the learner knows the set of possible slopes (it is not necessary for the learner to be told the slope of each individual box, just the set of possible values over all boxes). The learner for this class is exactly the learner for the axis-parallel case except that each box in the learner's hypothesis for a target in that class is replaced by one box for each of the possible slopes in this class. The remainder of the algorithm generalizes trivially. A learning algorithm for this class, efficient only for constant values of d, is given by Bshouty *et al.* [8].

7. VARIATIONS OF THE MODEL

In this section we briefly discuss several variants of our teaching model. The first of these allows the teacher to give multiple counterexamples to a single query. The next variant allows for incomplete or malicious membership queries.

As our model is defined the teacher provides the learner with a single counterexample to any query, paralleling the standard query learning model. Consider, however, a model in which the teacher can answer any query with a constant number of counterexamples. This, for example, would allow the randomized learner for DNF in Section 4.3 to ask a single equivalence query for each superset/subset query pair since the teacher could provide both a positive and a negative counterexample as an answer. It seems unlikely, however, that this change increases the power of the model.

The next change we consider to the model concerns allowing noise. Specifically we examine noise in the membership queries. Angluin and Slonim [4] introduced the model of incomplete membership queries in which any membership query can be answered "I don't know" independently at random. The only restriction is that the answers are persistent— the answer given for a query the first time it is asked is given every time it is asked. In this model, Angluin and Slonim showed that monotone DNF is learnable, with high probability, in polynomial time. It is easy to allow for this phenomenon in our model. We simply have the teacher flip a (possibly) biased coin and give the correct answer for the query if the coin is heads and answer "I don't know" if it is tails. It is clear that in this model we can make the same claims about learnability (with IMQ) implying teachability that we make in general. What is less clear is if the ability of the teacher to provide useful counterexamples to equivalence queries can help compensate for the noise (say by reducing the number of membership queries necessary) and thus allow the teachability of a class in this model that is not learnable with incomplete membership queries.

Finally, we consider malicious membership queries as introduced by Angluin and Krikis [3]. In this model the membership queries are answered *incorrectly* at the discretion of an omniscient adversary. As with incomplete membership queries the noise is persistent. The adversary has a bound of l on the number of instances on which it can lie and the learner is allowed time polynomial in l. (Sloan and Turán [27] introduced a similar model in which membership queries are answered "I don't know" at the discretion of an adversary.) This change is easily incorporated into our model since we can allow the adversary to change the answers to membership queries at its discretion with a

bound of l on the number of times this can be done. Again it is easy to see that we can teach in this model any class that can be learned with malicious membership queries but it is unclear if the model allows the teachability of a class that is not learnable.

8. CONCLUDING REMARKS

In this paper we have presented an interactive model of teaching that more accurately models the nature of the relationship between teachers and students. We have shown that any concept class that is learnable using example-based queries, even by a randomized learner, is teachable in this model. We have demonstrated the power of the model by showing that the class of DNF formulas is teachable using only equivalence queries. The learnability of this class is an important open problem.

An intriguing open problem is to try to find ${}_{\mathrm{T}}\mathrm{I}_{\mathrm{L}}$ pairs for concept classes that are representationally more powerful than DNF formulas. In particular, does there exist a deterministic ${}_{\mathrm{T}}\mathrm{I}_{\mathrm{L}}$ pair for polynomial size circuits? In a different direction, it would be interesting to pursue the power of randomization in the model. What classes can be learned with a randomized learner and teacher? Is there a probabilistic, polynomial time teacher for the class of DNF?

Another interesting research direction is to extend this model to work in the PAC sense. That is, change the requirement of the learner to return, with high probability, a good approximation of the target concept. This changes the relationship between the teacher and learner since the examples seen by the learner would be chosen according to an unknown probability distribution. One idea is to allow the teacher to first communicate to the learner some polynomial number of "good" examples that communicate an important aspect of the target.

APPENDIX: ATTEMPTING TO PREVENT COLLUSION

During development of this model, we attempted to prevent collusion between the teacher and learner. While all of the methods we tried were adversary based, each successive attempt increased in complexity. Each was also defeated. In this section we outline one method used by the teacher and learner to frustrate our attempts to prevent collusion. We hope that this discussion illustrates the difficulty of collusion prevention in an interactive teaching model.

The purpose of the adversary in the teaching protocol is to prevent collusion. While unsuccessful, we maintain the adversary because it can prevent some forms of collusion (e.g., inter-example collusion in the absence of intra-example collusion). It also seems that if a general collusion prevention scheme is possible it will be adversary-based.

We begin with the adversarial method currently used in our model; an omniscient adversary has the ability to add queries to those asked by the learner and to add answers to those provided by the teacher. We demonstrate a system implemented by the teacher and learner that allows collusion in the presence of this adversary. This construction is due to Angluin [2].

Let p be an $n/2$ bit prime. In polynomial time a randomized teacher can generate p with high probability. If the teacher is computationally unbounded then it can generate p with probability 1. The teacher can send p to the learner in a single example (without loss of generality, assume it is the first example sent to the learner). The learner then knows that one of the examples in the first answer set received represents p. By interleaving computations for the remainder of the teaching session with each candidate for p the learner will obtain the intended result. For the remainder of this discussion we assume that the learner knows p. We assume that the teacher and learner have agreed on some method, using residues mod p, to encode a hypothesis, h_t, equivalent to the target. That is, the teacher and learner agree on some encoding method. The teacher then breaks the encoding for h_t into pieces of length $n/2$ (each a residue mod p) to pass to the learner. Let r be the number of residues required to specify h_t. The goal of the teacher is to communicate to the learner the sequence of residues $\langle a_0, ..., a_{r-1} \rangle$.

The sequence is reconstructed by the learner in an iterative fashion, first building pairs $\langle a_0, a_1 \rangle$, $\langle a_2, a_3 \rangle$, ..., then quadruples $\langle a_0, a_1, a_2, a_3 \rangle$, ... and so on until the entire sequence has been constructed. The reason for iteratively building the sequence is to keep the total number of candidate sequences small (recall that the adversary is adding instances to confuse the learner). The way that this is accomplished is explained in a moment.

Each d-tuple of residues is encoded as a degree $d-1$ polynomial where each a_i is a coefficient. Each polynomial, q, is encoded by its value $q(0), q(1), ..., q(r-1)$. We use q_1 to represent the first pair, q_2 to represent the second pair, $q_{r/2+1}$ to represent the first quadruple, etc. The teacher sends the values in a round-robin manner: $q_1(0), q_2(0), ..., q_s(0), q_1(1), ..., q_2(1), ...$ Each $q_i(j)$ is represented in a single instance where the first $n/4$ bits represent i, the next $n/4$ bits represent j, and the last $n/2$ bits represent $q_i(j)$ mod p. During this stage of the algorithm the learner maintains two minimally consistent hypotheses: h_+ and h_-. h_+ is a list of the positive examples seen by the learner and h_- is a list of the negative examples seen by the learner. The learner simply alternates asking equivalence queries with h_+ and h_-, allowing the teacher to give as counterexamples any instance not yet seen by the learner.

We now consider the learner's strategy. According to the schedule described above, the teacher will attempt to send $(i, j, q_i(j) \bmod p)$ as the answer to query $2(js+i)$ or query $2(js+i)+1$ (since this instance is either positive or negative and will thus be appropriate as an answer to only one of the

equivalence queries EQ(h_+) or EQ(h_-)). Let $R_{i,j}$ denote the set of all instances received by the learner (those of the adversary in addition to those of the teacher) through query $2(js+i)+1$ that have "tag" (i,j). First, the learner constructs pairs of coefficients. Since each pair is represented as a degree one polynomial the learner can reconstruct each pair from two points. However, the learner does not know which points to use. By taking $R_{1,0} \times R_{1,1}$ (the sets of candidate points for polynomial q_1 evaluated at 0 and 1) the learner creates (after some arithmetical manipulation) $|R_{1,0}| \cdot |R_{1,1}|$ candidate pairs for (a_0, a_1). Call this set C_1. By looking at $R_{1,2}$ the learner can eliminate those candidates from C_1 that do not take on a value in $R_{1,2}$ at $j=2$. The learner can also eliminate from $R_{1,2}$ any values that do not correspond to one of the polynomials in C_1 at $j=2$. At this point it must be the case that either: at least two candidates in C_1 have the same value at $j=2$ or $|C_1| \leqslant |R_{1,2}|$. In the first case the learner continues checking $R_{1,j}$ until $|C_1| \leqslant |R_{1,k}|$ for some k. Since distinct lines can intersect in at most one point, this must happen for

$$k \leqslant \left(2 + \binom{|R_{1,0}| \cdot |R_{1,1}|}{2}\right).$$

The learner satisfies this condition independently for each pair (a_{2i}, a_{2i+1}). Then processing can begin on the quadruples. For example, if C_1 is the set of candidates for (a_0, a_1) and C_2 is the set of candidates for (a_2, a_3) then the cartesian product of C_1 and C_2 gives the set of candidates for the quadruple (a_0, a_1, a_2, a_3). Call this set of candidates C_t. Then the learner checks these candidates using values in the sets $R_{t,j}$ for $j=0, 1, 2, \ldots$, eliminating candidates from C_t until $|C_t| \leqslant |R_{t,q}|$ for some $q \geqslant 0$.

Building pairs into quadruples, quadrupes into octuples, etc., the learner eventually satisfies the condition for the entire sequence. That is, $|C_s| \leqslant |R_{s,y}|$. There are $|R_{u,v}| \cdot |R_{w,z}|$ candidates for the entire sequence, where $R_{u,v}$ and $R_{w,z}$ are sets of candidates for the sequences of length $r/2$. Thus, there are

$$\binom{|R_{u,v}| \cdot |R_{w,z}|}{2}$$

pairs of candidates. Since two degree-k polynomials can intersect in at most k points, each pair of candidates can intersect in at most $r-1$ points. Thus, $|C_s| \leqslant |R_{s,y}|$ for

$$y \leqslant \left((r-1) \cdot \binom{|R_{u,v}| \cdot |R_{w,z}|}{2}\right).$$

At this point the learner can make an equivalence query with each candidate in C_s one of which will be answered "yes."

The learner asks $O(rs)$ queries to build the $R_{i,j}$, where $r = O(\|h_t\|)$ and s, the number of polynomials passed, is at most $2r$. Thus, to build the candidates, the learner asks a number of queries polynomial in the size of the hypothesis being encoded. The number of queries asked at the end is polynomial in the number of examples (including adversarial examples) seen by the learner. The time used by the learner is polynomial as well.

In an effort to prevent such a scheme we allowed the adversary the additional ability of remapping the instance space by rearranging the boolean variables. This was intended to prevent the intra-example collusion that allows the above method to defeat the adversary—specifically, the use of the tag bits indicating i and j. The way this worked was that the adversary would change the learner's queries to an alternate boolean basis (the adversary could still add queries and answers). When the teacher answered these queries the adversary would remap the answers back to the original basis. Thus, the teacher could not pass the type of "tag" information necessary. However, as its first query the learner could pass a set of examples that illustrated for the teacher what its basis was (*i.e.*, for $n=5$ show the instances 10000, 11000, 11100, 11110). The adversary would then remap this query to another basis. But only 10000 would map to an instance with only one 1. This would tell the teacher which bit y_1 mapped to. Then, with this knowledge, the teacher could discover where y_2 was mapped by looking at the only instance in the query with two 1s. Even after adding of queries by the adversary the teacher would know that the basis was one of a polynomial number of candidates and could work with them in a round-robin fashion. Note that remapping by using bitwise *XOR* with some string in $\{0,1\}^n$ (as in Bshouty's monotone theory) is even easier to circumvent. Other types of remappings destroy structure in the lattice that is essential to some classes (such as monotone classes). Once the basis is known to the teacher, a scheme as above could be used for encoding. Thus, the teacher and learner can bypass this additional method as well.

ACKNOWLEDGMENTS

We thank Sally Goldman for her many suggestions and her comments on previous drafts of this paper and Dana Angluin and Ken Goldman for useful suggestions relevant to this work. We also thank the COLT '95 program committee.

REFERENCES

1. D. Angluin, Queries and concept learning, *Mach. Learning* **2**, No. 4 (1988), 319–342.
2. D. Angluin, personal communication, 1994.
3. D. Angluin and M. Krikis, Learning with malicious membership queries and exceptions, *in* "Proceedings, Seventh Annual ACM Conference on Computational Learning Theory, July 1994," pp. 57–66.

4. D. Angluin and D. K. Slonim, Randomly fallible teachers: Learning monotone DNF with an incomplete membership oracle, *Mach. Learning* **14**, No. 1 (1994), 7–26.
5. M. Anthony, G. Brightwell, D. Cohen, and J. Shawe-Taylor, On exact specification by examples, *in* "Proceedings, Fifth Annual Workshop on Computational Learning Theory, July 1992," pp. 311–318.
6. N. H. Bshouty, Exact learning via the monotone theory, *in* "34th Annual Symposium on Foundations of Computer Science, November 1993."
7. N. H. Bshouty, R. Cleve, S. Kannan, and C. Tamon, Oracles and queries that are sufficient for exact learning, *in* "Proceedings, Seventh Annual ACM Conference on Computational Learning Theory, July 1994," pp. 130–139.
8. N. H. Bshouty, P. W. Goldberg, S. A. Goldman, and H. D. Mathias, "Exact Learning of Discretized Concepts," Technical Report WUCS-94-19, Washington University, 1994.
9. R. Freivalds, E. Kinber, and R. Wiehagen, Inference from good examples, *Theoretical Computer Science* **110** (1993), 131–144.
10. P. W. Goldberg, S. A. Goldman, and H. D. Mathias, Learning unions of boxes with membership and equivalence queries, *in* "Proceedings, Seventh Annual ACM Conference on Computational Learning Theory, July 1993."
11. S. Goldman and D. Mathias, Teaching a smarter learner, *in* "Proceedings, Sixth Annual ACM Conference on Computational Learning Theory," pp. 67–76, ACM Press, New York, NY, 1993. To appear, *J Comput. System Sci.*.
12. S. A. Goldman and M. J. Kearns, On the complexity of teaching, *J. Comput. System Sci.* **50**(1) (1995), 20–31.
13. S. A. Goldman, R. L. Rivest, and R. E. Schapire, Learning binary relations and total orders, *SIAM Journal on Computing* **22**(5) (1993), 1006–1034.
14. S. A. Goldman and R. H. Sloan, The power of self-directed learning, *Mach. Learning* **14**(3) (1994), 217–294.
15. T. Hegedűs, Combinatorial results on the complexity of teaching and learning, *in* "Proceddings, 19th International Symposium on Mathematical Foundations of Computer Scinece," Springer-Verlag, New York/Berlin, 1994.
16. T. Hegedűs, Generalized teaching dimensions and the query complexity of learning, *in* "Proceedings, Eighth Annual ACM Conference on Computational Learning Theory, July 1995."
17. J. Jackson, An efficient membership-query algorithm for learning DNF with respect to the uniform distribution, *in* "35th Annual Symposium on Foundations of Computer Science, November 1994," pp. 42–53.
18. J. Jackson and A. Tomkins, A computational model of teaching, *in* "Proceedings, Fifth Annual Workshop on Computational Learning Theory," pp. 319–326, ACM Press, New York, 1992.
19. M. J. Kearns, "The Computational Complexity of Machine Learning," MIT Press, Cambridge, MA, 1990.
20. S. Lange and R. Wiehagen, Polynomial-time inference of arbitrary pattern languages, *New Generation Computing* **8** (1991), 361–370.
21. B. K. Natarajan, On learning Boolean functions, *in* "Proceedings, Nineteenth Annual ACM Symposium on Theory of Computing, May 1987," pp. 296–304.
22. R. L. Rivest and Y. L. Yin, Being taught can be faster than asking questions, *in* "Proceedings, Eighth Annual ACM Conference on Computational Learning Theory, July 1995."
23. K. Romanik, Approximate testing and learnability, *in* "Proceedings, Fifth Annual Workshop on Computational Learning Theory," pp. 327–332, ACM Press, New York, 1992.
24. K. Romanik and C. Smith, "Testing Geometric Objects," Technical Report UMIACS-TR-90-69, University of Maryland College Park, Department of Computer Science, 1990.
25. S. Salzberg, A. Delcher, D. Heath, and S. Kasif, Learning with a helpful teacher, *in* "12th International Joint Conference on Artificial Intelligence, August 1991," pp. 705–711.
26. A. Shinohara and S. Miyano, Teachability in computational learning, *New Generation Computing* **8** (1991), 337–347.
27. R. H. Sloan and G. Turán, Learning with queries and incomplete information, *in* "Proceedings, Seventh Annual ACM Conference on Computational Learning Theory, July 1994," pp. 237–245.

4. D. Angluin and D. K. Slonim, Randomly fallible teachers: Learning monotone DNF with an incomplete membership oracle, *Mach. Learning* **14**, No. 1 (1994), 7–26.
5. M. Anthony, G. Brightwell, D. Cohen, and J. Shawe-Taylor, On exact specification by examples, *in* "Proceedings, Fifth Annual Workshop on Computational Learning Theory, July 1992," pp. 311–318.
6. N. H. Bshouty, Exact learning via the monotone theory, *in* "34th Annual Symposium on Foundations of Computer Science, November 1993."
7. N. H. Bshouty, R. Cleve, S. Kannan, and C. Tamon, Oracles and queries that are sufficient for exact learning, *in* "Proceedings, Seventh Annual ACM Conference on Computational Learning Theory, July 1994," pp. 130–139.
8. N. H. Bshouty, P. W. Goldberg, S. A. Goldman, and H. D. Mathias, "Exact Learning of Discretized Concepts," Technical Report WUCS-94-19, Washington University, 1994.
9. R. Freivalds, E. Kinber, and R. Wiehagen, Inference from good examples, *Theoretical Computer Science* **110** (1993), 131–144.
10. P. W. Goldberg, S. A. Goldman, and H. D. Mathias, Learning unions of boxes with membership and equivalence queries, *in* "Proceedings, Seventh Annual ACM Conference on Computational Learning Theory, July 1993."
11. S. Goldman and D. Mathias, Teaching a smarter learner, *in* "Proceedings, Sixth Annual ACM Conference on Computational Learning Theory," pp. 67–76, ACM Press, New York, NY, 1993. To appear, *J Comput. System Sci.*.
12. S. A. Goldman and M. J. Kearns, On the complexity of teaching, *J. Comput. System Sci.* **50**(1) (1995), 20–31.
13. S. A. Goldman, R. L. Rivest, and R. E. Schapire, Learning binary relations and total orders, *SIAM Journal on Computing* **22**(5) (1993), 1006–1034.
14. S. A. Goldman and R. H. Sloan, The power of self-directed learning, *Mach. Learning* **14**(3) (1994), 217–294.
15. T. Hegedűs, Combinatorial results on the complexity of teaching and learning, *in* "Proceddings, 19th International Symposium on Mathematical Foundations of Computer Scinece," Springer-Verlag, New York/Berlin, 1994.
16. T. Hegedűs, Generalized teaching dimensions and the query complexity of learning, *in* "Proceedings, Eighth Annual ACM Conference on Computational Learning Theory, July 1995."
17. J. Jackson, An efficient membership-query algorithm for learning DNF with respect to the uniform distribution, *in* "35th Annual Symposium on Foundations of Computer Science, November 1994," pp. 42–53.
18. J. Jackson and A. Tomkins, A computational model of teaching, *in* "Proceedings, Fifth Annual Workshop on Computational Learning Theory," pp. 319–326, ACM Press, New York, 1992.
19. M. J. Kearns, "The Computational Complexity of Machine Learning," MIT Press, Cambridge, MA, 1990.
20. S. Lange and R. Wiehagen, Polynomial-time inference of arbitrary pattern languages, *New Generation Computing* **8** (1991), 361–370.
21. B. K. Natarajan, On learning Boolean functions, *in* "Proceedings, Nineteenth Annual ACM Symposium on Theory of Computing, May 1987," pp. 296–304.
22. R. L. Rivest and Y. L. Yin, Being taught can be faster than asking questions, *in* "Proceedings, Eighth Annual ACM Conference on Computational Learning Theory, July 1995."
23. K. Romanik, Approximate testing and learnability, *in* "Proceedings, Fifth Annual Workshop on Computational Learning Theory," pp. 327–332, ACM Press, New York, 1992.
24. K. Romanik and C. Smith, "Testing Geometric Objects," Technical Report UMIACS-TR-90-69, University of Maryland College Park, Department of Computer Science, 1990.
25. S. Salzberg, A. Delcher, D. Heath, and S. Kasif, Learning with a helpful teacher, *in* "12th International Joint Conference on Artificial Intelligence, August 1991," pp. 705–711.
26. A. Shinohara and S. Miyano, Teachability in computational learning, *New Generation Computing* **8** (1991), 337–347.
27. R. H. Sloan and G. Turán, Learning with queries and incomplete information, *in* "Proceedings, Seventh Annual ACM Conference on Computational Learning Theory, July 1994," pp. 237–245.