Machine Learning, 25, 151–193 (1996)

CLASSIC Learning

MICHAEL FRAZIER* frazier@cs.acu.edu
Computer Science Department, Abilene Christian University, Abilene, TX 79699

LEONARD PITT** pitt@cs.uiuc.edu
Department of Computer Science, University of Illinois at Urbana-Champaign
1304 W. Springfield Avenue, Urbana, IL 61801

Editor: Thomas Hancock

Abstract. *Description logics*, also called *terminological logics*, are commonly used in knowledge-based systems to describe objects and their relationships. We investigate the learnability of a typical description logic, CLASSIC, and show that CLASSIC sentences are learnable in polynomial time in the exact learning model using equivalence queries and membership queries (which are in essence, "subsumption queries"—we show a prediction hardness result for the more traditional membership queries that convey information about specific individuals).

We show that membership queries alone are insufficient for polynomial time learning of CLASSIC sentences. Combined with earlier negative results (Cohen & Hirsh, 1994a) showing that, given standard complexity theoretic assumptions, equivalence queries alone are insufficient (or random examples alone in the PAC setting are insufficient), this shows that both sources of information are necessary for efficient learning in that neither type alone is sufficient. In addition, we show that a modification of the algorithm deals robustly with persistent malicious two-sided classification noise in the membership queries with the probability of a misclassification bounded below 1/2. Other extensions are considered.

Keywords: Description logic, polynomial-time learning, CLASSIC, subsumption, queries, knowledge acquisition

1. Introduction

We address the problem of efficient knowledge acquisition from the vantage point of computational learning theory. Traditionally, computational learning theory has focused on propositional domains. We investigate learning in the first-order domain of *description logics* or *terminological logics*. Specifically we consider the learnability of the description logic known as CLASSIC (Borgida et al., 1989). To the extent that CLASSIC is a typical description logic, our results generalize to a variety of other such logics.

Description logics are more expressive than the propositional calculus. A description logic statement is essentially a first-order predicate calculus formula in which all but one variable is quantified. Therefore, the meaning of a statement in a description logic, instead of being either true or false for a given interpretation, is the subset of the universe satisfying the statement. For example, suppose that the universe is a set of dogs, `brown`(x) asserts that x is brown, and `smaller`(x,y) asserts that y is smaller than x. If it happens to be the case that `Rex` is the only shaggy dog and `Fido` is the only brown dog, then $(\forall y)$ `brown`$(x) \wedge$ `smaller`(x, y) is a well-formed description logic statement denoting

* Supported in part by NSF Grant IRI-9014840, and by NASA grant NAG 1-613. This work was completed while at the University of Illinois at Urbana-Champaign, and at Southern University at New Orleans.
** Supported in part by NSF grant IRI-9014840.

the set $\{\texttt{Fido}\}$ provided `Fido` is the largest dog in the universe; otherwise the empty set is denoted. Likewise $\texttt{brown}(x) \wedge \texttt{shaggy}(x)$ denotes the empty set. Neither statement is a closed formula in the predicate calculus and neither statement has an associated truth value. Thus, description logics have a different flavor than the predicate calculus. Description logics comprise natural classes of formulas; not only are they the subject of theoretical investigation within the field of knowledge representation, but they also find use in practical knowledge-based systems (Beck et al., 1989; Borgida, 1992; Borgida & Patel-Schneider, 1992; Brachman et al., 1983; Cohen & Hirsh, 1994b; Devanbu et al., 1991; Mays et al., 1987; Patel-Schneider, 1989).

1.1. CLASSIC

CLASSIC permits constructing certain quantified descriptions that distinguish a particular subset of a domain I of *individuals*. CLASSIC descriptions contain *primitive symbols* which get mapped to arbitrary subsets of I, *disjoint primitive symbols* which get mapped to mutually disjoint subsets of I, *roles* which get mapped to binary relations on I, and *attributes* which are roles that happen to be functions. Further, CLASSIC sentences contain constructors which manipulate these primitives, disjoint primitives, roles, and attributes, in order to permit the denotation of complicated subsets of I. The following synopsis and semantics of CLASSIC is excerpted from a variety of sources (Borgida & Patel-Schneider, 1992; Cohen et al., 1992; Cohen & Hirsh, 1994a).

(SAME-AS ($r_{1,1} \ldots r_{1,k_1}$) ($r_{2,1} \ldots r_{2,k_2}$)) denotes the set of individuals
$\{x : r_{1,k_1}(\cdots(r_{1,2}(r_{1,1}(x)))) = r_{2,k_2}(\cdots(r_{2,2}(r_{2,1}(x))))\}$
for which composing the first chain of attributes is the same as composing the second chain of attributes.

(ALL r D) denotes the set $\{x : \forall y \ [r(x,y) \rightarrow D(y)]\}$ of individuals for which *all* of the r-related individuals satisfy CLASSIC description D.

(AND $D_1 \ldots D_n$) denotes the set $\{x : D_1(x) \wedge \cdots \wedge D_n(x)\}$ of individuals that satisfy each CLASSIC description $D_1, \ldots, D_n$.

(AT-LEAST n r) denotes the set $\{x : |\{y : r(x,y)\}| \geq n\}$ of individuals having at least n r-related individuals.

(AT-MOST n r) denotes the set $\{x : |\{y : r(x,y)\}| \leq n\}$ of individuals having at most n r-related individuals.

(PRIM p_i) denotes the subset of individuals denoted by the primitive symbol p_i (provided by the interpretation). (In our illustrations we omit this formalism and use descriptions such as `brown` to denote the primitive set of things which are brown.)

(FILLS r $p_1 \ldots p_n$) denotes the set $\{x : \exists y_i \in p_i$ such that $r(x,y_1) \wedge \cdots \wedge r(x,y_n)\}$, where the p_i are disjoint primitive symbols.

(ONE-OF $p_1 \ldots p_n$) denotes the set $\cup_{i=1}^{n} p_i$, where the p_i are disjoint primitive symbols.

Descriptions are built from the individuals, primitives, and other descriptions. For example if our set of individuals is the set of all dogs and breeds[1] and we have at our disposal the primitive concept `brown` for the set of brown dogs, the role `smaller` for comparing dog sizes, the attribute `breed` for denoting breeds, the attribute `father` for associating a dog with its father, the attribute `mother` for associating a dog with its mother, and the role `classmate` denoting the obedience school classmates, then if we wished to denote the set

$$\{x : \forall y\ \texttt{classmate}(x, y) \rightarrow [\texttt{brown}(y) \wedge |\{z : \texttt{smaller}(y, z)\}| \geq 20 \wedge \texttt{breed}(\texttt{mother}(y)) = \texttt{breed}(\texttt{father}(\texttt{father}(y)))\ ]\ \}$$

of dogs *all* of whose obedience school classmates were brown, larger than at least twenty other dogs, and had mother and paternal grandfather of the same breed, we would write:

```
(ALL classmate (AND brown
                    (AT-LEAST 20 smaller)
                    (SAME-AS (mother breed) (father father breed))))
```

1.2. CLASSIC *Semantics and the Learning Problem*

The meaning of a description logic statement depends on a particular interpretation. It is a set selector: Given a choice of a universal set of individuals I, an assignment of the primitive symbols to subsets of I, an assignment of the roles to binary relations on I, and an assignment of attributes to functions from I to I, the statement denotes the set of elements x in I that cause the corresponding first order expression to evaluate to true, given the semantics above.

For example, a reasonable definition of **Tired-people** might be the set of people who have at least one child. In CLASSIC we would write the sentence S_1 = (**AT-LEAST** 1 `Child`), where `Child` is a role (binary relation). Now consider the interpretation described by the relational database given in Figure 1.

The universe I of all individuals is understood to be the set of all individuals appearing ({person1, ..., person6}) in any of the relations. The primitive subsets are `Blonde` and `Red-head`, there is a single function symbol (attribute) `Mother`, indicating, for example, that `Mother`(person1) = person2, and there is a single role that is not an attribute, the relation `Child`, indicating, for example, that the children of person3 are exactly person2 and person5.

The denotation of the sentence S_1 above, given the interpretation of Figure 1, is exactly the set of individuals {person2, person3, person5}.

Two CLASSIC sentences are said to be equivalent if they have the same denotation regardless of the interpretation (that is, if they pick out the same subset of the domain regardless of what "world" we are in).

Now consider the CLASSIC sentence S_2 = (**ALL** `Mother Red-head`), which describes the set of individuals x such that all mothers of x (there is only one, since `Mother` is a

Blonde
person1
person2
person6

Red-head
person3
person4

Mother	
person1	person2
person2	person3
person3	person4
person4	person6
person5	person3

Child	
person2	person1
person3	person2
person3	person5
person5	person6

Figure 1. Relational database describing individuals. The database contains unary relations `Blonde` and `Red-head`, and binary relations `Mother` and `Child`.

function) happen to be a red-head. In short, this describes people with red-headed mothers. For the interpretation given by Figure 1, S_2 denotes exactly the set of individuals {person2, person3, person5}. Thus, S_1 and S_2 have identical denotations for the particular interpretation given in the figure, but in general S_1 and S_2 are not logically equivalent, because it is easy to construct an interpretation for which the set of people with red-headed mothers is not the same as the set of people having at least one child.

The fact that inequivalent CLASSIC sentences can have identical denotations when restricted to particular interpretations suggests at least two possibilities for modeling the learning of CLASSIC sentences: *Learning from Individuals*, and *Learning General Descriptions*.

Learning from Individuals: If we adopt the view that there is only one "world", given by a particular interpretation, then it should not matter to us whether two inequivalent CLASSIC sentences have the same denotation with respect to the (one and only) world. We should be equally happy with the sentence S_2 = (**ALL** `Mother Red-head`) to describe the set of tired people, as with the intuitively more meaningful one given by S_1. This view suggests that any description that picks out the right set of individuals should suffice. Indeed, all such descriptions are semantically indistinguishable unless we admit either the possibility of other worlds/interpretations, or unless we assign them some additional meaning outside of our formal system. From this vantage point, the particular subset of individuals embodies exactly the concept to be learned.

In the various accepted models of inductive concept learning from examples (which we will more properly introduce below), an unknown *target concept* is to be inferred by a learning agent. A concept is simply a subset of some domain, which cleaves the domain into *positive examples* (those in the concept), and *negative examples* (those not in the concept).

Successful learning is typically defined to be that of identifying the target concept (or one "close" to it), in a computationally efficient manner, given information about which domain elements are positive examples and which are negative examples.

In light of the above discussion, a natural way to define the learning problem for CLASSIC descriptions would be as follows. A target CLASSIC description is chosen. There is also a fixed, known interpretation. Positive examples are individuals in the domain of the interpretation that are in the denotation of the target concept. Negative examples are individuals that are not in the denotation. Successful inference requires not that a logically equivalent (or "approximately logically equivalent") CLASSIC sentence be found, but rather that the learning agent find any CLASSIC sentence that has the same denotation (or approximately the same denotation) *on the given interpretation*.

One question that arises is exactly how the interpretation is given to the learning algorithm as input. If the domain over which the interpretation is defined (i.e., the set of all possible examples) is small (poly-sized), then the target concept can be inferred trivially by simply saving all positive examples — an uninteresting learning problem results. On the other hand, if the domain is large, then if the entire interpretation is given to the learning algorithm, and the algorithm is allowed time that depends polynomially on the size of the database describing the interpretation, then this would allow perhaps exponential time for the learning task.

Alternatively, suppose that the interpretation, represented by an exponentially sized (number of tuples) database over some polynomially sized number of primitive symbols, relation symbols, and function symbols, is *not* given explicitly. Instead, with each (positive or negative) example individual, the learning algorithm is provided complete information about the individual (i.e., all relations which the individual participates in, the value of all functions, etc.) Because of the possibility of "function-chaining", such complete information about an individual might well necessitate providing the entire database. Indeed, since the target concept will specify a set of individuals not only by the properties they possess, but also by the relationships that they have with other individuals, if the learning algorithm did not have at least potential access to the entire interpretation, then learning might be impossible.

It appears that the most reasonable alternative is to assume that a very large database representing the world is available, but not explicitly input to the learner. The learner is instead allowed to make database queries to determine relationships about various individuals. Positive (respectively, negative) examples are then just keys of individuals that are denoted (respectively, not denoted), by the target concept. It is the task of the learning agent to use the database effectively in order to extract the common properties of the positive-example-individuals that are not shared by the negative-example-individuals, and to express this information in the form of a CLASSIC sentence.

We will consider this approach in Section 6, in which we show that even fairly constrained CLASSIC sentences cannot be learned from individuals in this manner, given well-accepted cryptographic assumptions.

Learning General Descriptions: One of the disadvantages of the model of learning from individuals discussed above is the implicit assumption that all information about each individual is available from the outset. It would appear that in the real world, we denote individual objects by describing them *sufficiently well* to distinguish them from others.

However, the threshold of what constitutes a sufficient discriminating description may well change in different contexts, and may evolve to include more information as either our need for more accurate discrimination changes, or as the changing environment renders insufficient a previously adequate discriminator. As a simple example, a dog breeder may find that "the black Labrador retriever" suffices to distinguish Fido from the rest of the animals at the kennel. However, if a new black Labrador is born, the description of Fido might change to "the adult black Labrador retriever". Similarly, if another adult black Labrador is acquired, then the description might be refined further.

Viewed this way, an individual is denoted by a *description itself*, as opposed to some unique key tied to a particular database. A description of an individual then really denotes a set of possible individuals, where we assume that the description is sufficiently specific so as to be unambiguous given the current environment and task at hand.

What language should be used to describe an individual? We employ the commonly used "single representation trick" — wherein the description of an individual is itself a CLASSIC sentence. This approach is supported by the description logic community (Borgida, 1992; Bobrow & Winograd, 1977; Dietterich et al., 1982), in which it is often convenient and desirable to represent concepts and examples using the same language. In fact, Cohen and Hirsh (1994a) note that in many implemented description logics, "it is possible to attach an arbitrary description to an instance [example], hence the distinction between instances [examples] and concepts is blurred."

If individuals are CLASSIC descriptions themselves, then for each interpretation they denote a particular subset of the domain over which the interpretation is defined. Note that a description, even though intended to be a description of an individual, will not necessarily denote a unique domain element for every possible interpretation. Consequently, the distinction between such descriptions, and arbitrary CLASSIC descriptions is lost.

If we are given a target CLASSIC description, which other CLASSIC descriptions then are positive examples, and which are negative examples? Again, following work in the description logic community (Cohen & Hirsh, 1994a; Borgida, 1992; Bobrow & Winograd, 1977; Dietterich et al., 1982), we define a positive example to be any CLASSIC description that denotes, for every possible interpretation, a subset of those individuals denoted by the target description. Thus, each positive example has denotation that is a subset of the denotation of the target concept, regardless of the interpretation. If sentence C is a positive example of sentence C_*, then we say that C_* *subsumes* C, because it has a larger denotation than C. This and similar viewpoints are also supported by previous work in inductive logic programming (Džeroski et al., 1992; Frazier & Page, 1993; Muggleton, 1991; Page & Frisch, 1992) and learning from entailment where positive examples of an unknown formula are clauses or other formulas that are entailed by the unknown formula (Angluin, 1988a; Angluin, 1988c; Frazier & Pitt, 1993).

Besides allowing for flexibility in representation of objects, this approach also has another advantage over the approach of learning from individuals. We have noted that the distinction between inequivalent CLASSIC sentences can be lost when restricted to a single interpretation. A perhaps more interesting view is that there is some general knowledge that we would like to acquire, not about one particular domain or interpretation, but about every possible one, and that this knowledge is realized by an unknown CLASSIC description. The

coincidental knowledge that people with at least one child all happen to have red-haired mothers in the world described by Figure 1 is not likely to be transportable to other environments. If, from the practical perspective, we view interpretations as particular databases representing real-world information, then it seems appropriate to demand that the general descriptions we learn should be applicable to all such interpretations. The approach of learning general descriptions demands that a CLASSIC description be found that classifies examples as positive or negative in the same way that the target CLASSIC sentence does. Because examples are CLASSIC descriptions, and positive examples must be subsumed by the target description, it follows that the learned description must be equivalent (have the same denotation for all possible interpretations) to the target description. Our main focus is this latter viewpoint of learning general descriptions. Below we summarize our results, and describe the learning protocols used.

1.3. Learning Protocols

We will employ two standard learning protocols: that of exactly learning from equivalence and membership queries (Angluin, 1988b), and that of PAC learning (Valiant, 1984; Blumer et al., 1989) with membership queries.

In both protocols, the learning algorithm may pose a *membership query*, which is an example x. In response, the learner is told whether or not x is a positive or negative example of the concept to be learned. Keeping in mind that in our setting, examples are themselves CLASSIC descriptions, a membership query is a CLASSIC sentence C, and the response is "yes" exactly when the target sentence C_* subsumes C. Our algorithm will take (perhaps unfair) advantage of the fact that such queries C may be arbitrary concepts, so perhaps it is more appropriate to call these subsumption queries, or even subset queries. As mentioned earlier, such distinctions are lost given the use of the single representation trick.

Because the subsumption relation for CLASSIC sentences is computable in polynomial time (Cohen et al., 1992; Cohen & Hirsh, 1994a), membership queries are efficiently computable by a teacher. Consequently, if membership queries model a type of active learning by asking questions of a teacher or domain expert, then such queries here should not prove to be computationally difficult for a reasonable teacher.

Exact learning with equivalence and membership queries: In addition to membership queries, in the exact learning model the algorithm may also conjecture any CLASSIC description H, and is told in response whether or not H is equivalent to the target description C_*. If H is not equivalent to C_*, then the algorithm receives a counterexample, which is a positive example of one of H and C_*, but not both. (I.e., a description C' that is subsumed by one but not the other.) We note that equivalence queries can also be answered in polynomial time by a teacher: C_* is equivalent to H if and only if each subsumes the other. If this is not the case, then the (at least) one that is not subsumed by the other is a counterexample, as it is trivially subsumed by itself, hence is a positive example of itself, but not of the other.

Because the size of counterexamples may vary, and are not under the direct control of the learning algorithm, there are technical subtleties in defining the appropriate model for

exact learning in polynomial time (Angluin, 1987). We follow the standard convention and require that for efficient learning, at any point during the run of our algorithm, the time used up to that point must be polynomial in the longest counterexample seen *so far*. Henceforth, we will simply say "in time polynomial in the longest counterexample" to mean this stronger statement.

A learning algorithm is said to learn CLASSIC from equivalence and membership queries if for any unknown target description C_*, the algorithm, after time polynomial in the length of the description C_*, and in the length of the longest counterexample, using only equivalence and membership queries, outputs a CLASSIC sentence H such that H and C_* are equivalent.

Standard transformations may be employed to obtain an algorithm that learns in the "PAC" learning model (Valiant, 1984) augmented with membership queries (described below), or in the on-line mistake bounded learning model (Angluin, 1988b; Littlestone, 1988) with membership queries.

PAC learning from random examples and membership queries: The model of PAC ("probably approximately correct") learning assumes that there is an arbitrary, fixed, unknown probability distribution defined over the set of positive and negative examples of size at most n. A learning algorithm draws random examples according to the distribution, with each example labeled as positive or negative according to an unknown target concept C_*. PAC learning requires that a learning algorithm output in polynomial time a concept H that, with probability at least $1 - \delta$, has error at most ϵ in classifying random examples as the target C_* does, where the error is measured with respect to the unknown distribution (Valiant, 1984).

We consider the PAC learning model where the algorithm may ask membership queries in addition to receiving randomly generated examples. More formally, a learning algorithm A is said to PAC-learn CLASSIC in polynomial time from random examples and membership queries if the following holds:

1. A receives as input parameters n, s, ϵ, and δ.
2. A outputs a CLASSIC sentence H within time polynomial in $n, s, \frac{1}{\epsilon}, \frac{1}{\delta}$.
3. A may make membership queries, or obtain upon request an example generated according to a fixed distribution D on example CLASSIC sentences of length at most n, and labeled according to some unknown target CLASSIC sentence C_* of representation length at most s.
4. For every possible n, distribution D, target concept C_*, upper bound s on $|C_*|$, and parameters ϵ, and δ, the output sentence H of A satisfies Prob$[D(C_* \triangle H) > \epsilon] < \delta$, where "$\triangle$" denotes symmetric difference.

A slightly relaxed model of PAC learning is that of polynomial *PAC-prediction*, where no syntactic requirements are placed on the hypothesis output by the learning algorithm (e.g., that it be a CLASSIC description), other than that it be a polynomial-time executable program that classifies examples well (Haussler et al., 1994; see also Pitt & Warmuth, 1990). In particular, we say that CLASSIC is PAC-predictable if conditions 1 through 4

hold above, but with condition 2 modified to read "A outputs a polynomial time program H within..." That this is called "prediction", as opposed to "learning" rests on the fact that the model is equivalent to one in which the algorithm need make no hypothesis whatsoever, but instead simply reach a state from which it can classify randomly generated unlabeled examples within some specified accuracy bound ϵ.

1.4. Summary of Results

We obtain both positive and negative results. On the positive side, we show that CLASSIC sentences can be learned with equivalence and membership queries, that they can be learned even in the presence of a high malicious misclassification noise rate in the membership query responses, and that a simple type of "weak" union of CLASSIC sentences can be learned. On the negative side, we show that predicting CLASSIC sentences from individuals is as hard as predicting arbitrary polynomial-sized circuits, and that membership queries alone do not suffice for learning CLASSIC.

The rest of this paper is organized as follows. After reviewing related work in Section 1.5, we show in Sections 2-4 that CLASSIC is exactly learnable in time polynomial in the size of the target description and the length of the longest counterexample, using membership (subsumption) and equivalence queries.

In Section 5 we argue that any algorithm using membership (subsumption) queries alone requires a number of queries that is exponential in the size of the target concept. Thus the positive result does not come solely from the membership queries. Cohen and Hirsh (1994a) showed that CLASSIC is not learnable in polynomial time (without membership queries) in the PAC model (assuming RP $\neq$ NP), hence CLASSIC cannot be learned from equivalence queries alone given the same assumption. Thus, neither membership nor equivalence queries are dispensable – they form a minimal set of learning queries for CLASSIC.

Section 6 addresses the problem of learning CLASSIC descriptions from individuals as described above. We show that if CLASSIC is PAC-predictable from individuals, then arbitrary Boolean circuits are PAC-predictable. Similarly, if CLASSIC can be PAC-predicted from individuals, by an algorithm that may also make membership queries (from a fixed database of possible examples), then Boolean circuits are PAC-predictable by an algorithm that also uses membership queries. It follows that, assuming the existence of one-way functions, CLASSIC cannot be learned from individuals, with or without (a certain type of) membership query.

In Section 7, we show that for one of two possible definitions of the semantics of a union of CLASSIC sentences, such unions can be learned exactly in polynomial time using equivalence and membership queries.

Finally, in Section 8, we consider a modification of our algorithm which demonstrates that CLASSIC remains learnable in polynomial time in the PAC learning model with membership queries, even when each membership query may be answered incorrectly by a malicious adversary with probability $\frac{1}{2} - \frac{1}{r}$, where r is any polynomial function of the size of the target concept. (The errors are *persistent*, so that the algorithm may not benefit from repeatedly asking the same question.) To our knowledge, this is the first algorithm for any concept class capable of coping robustly with such errors.

1.5. Comparison to Previous Results

Automating propositional concept discovery has been well studied (Angluin, 1992). In comparison, efficient first-order learnability has been less well studied. Even so, some results are known. Cohen (1993b) gives a PAC learning algorithm for function-free, two clause, closed, linearly recursive, ij-determinant logic programs; he also shows (Cohen, 1993a) that when the condition of linear recursiveness is relaxed, the learning problem becomes cryptographically hard. Page and Frisch (1992) show that constrained atoms (a typed logic) are efficiently learnable. Frazier and Page (1993) provide a learning algorithm for a syntactically restricted subclass of first-order Horn formulas Džeroski et al. (1992) provide a learning algorithm for a different restriction of first-order Horn formulas. A small number of other results, especially those related to Inductive Logic Programming, can be found in the survey by Cohen and Page (1994).

Haussler (1989) investigated existentially quantified conjunctive concepts and described a graph representation for those concepts. He showed that learning some very simple scene descriptions is difficult. Specifically, he showed that even restricted to unary atoms, such concepts are not learnable from random examples unless *RP = NP*, but did give a learning algorithm for settings where the algorithm may use a richer vocabulary than that from which the target was chosen. Indeed, positive first-order learning results appear to be quite rare for "natural" classes of first-order formulas. It would seem that the difficulty of the learning task he faced revolved around the ambiguity admitted by the graphical representation required to capture existential quantification in the concept class he investigated; our concept class does not permit existential quantification. It will be seen that the graphs we use suffer no such ambiguity, thus we are able to avoid the difficulty he faced.

The work most closely related is that of Cohen and Hirsh (1994a) who employ a graphical representation developed by Borgida and Patel-Schneider (1992) for CLASSIC concepts. To explain their results, and present ours, we briefly explain the notion of a *labeled equivalence graph* (called a *concept graph* in related work of Borgida and Patel-Schneider (1992), Cohen et al. (1992), and Cohen and Hirsh (1994a).

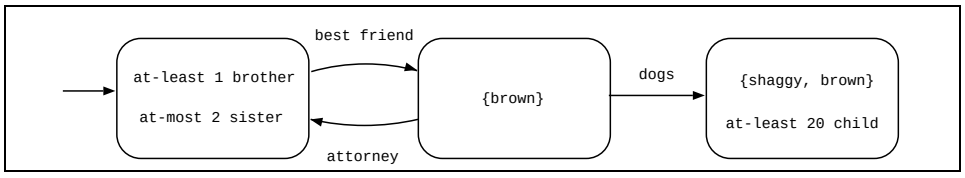

Figure 2. A labeled equivalence graph.

Consider the graph in Figure 2. This is a graphical depiction of the CLASSIC description of the set of individuals who have at least one brother and at most two sisters, whose best friend has brown hair, who are their best friend's attorney, and whose best friend only has brown, shaggy dogs that have at least twenty puppies. The cycle in the graph also asserts infinitely many other SAME-AS conditions – for example, conditions about the best friend's attorney's best friend.

Formally defined in Section 2, a labeled equivalence graph is a rooted, directed, vertex- and edge-labeled graph. Further, no vertex has two identically labeled outgoing edges. The edge labels represent binary relations over the universe of individuals, and an edge demands that all individuals in the image of the relation satisfy the constraints asserted by the vertex to which the edge leads. Those labeled equivalence graphs that correspond to CLASSIC descriptions also satisfy the following additional property: Any pair of edge-disjoint directed paths between a pair of vertices involve only binary relations which are in fact functions – this pair asserts that the individual selected along one path must be the same as the individual selected along the other path. This restriction is apparently a necessary one in order to allow for tractable subsumption. We associate a subset of the set of individuals I with each vertex in the graph — the subset of individuals satisfying every constraint (whether vertex label, edge label, or assertion of equivalence at some reachable vertex) asserted by the graph. The set of individuals denoted by the graph is exactly the set of individuals associated with the root. Note that the presence of directed cycles in the graph, and in particular, those involving the root, implies that the root concept is being defined in terms of other concepts, . . . , which are in turn being defined in terms of the root. Thus, cycles allow co-referential definitions.

Polynomial-time algorithms exist for transforming any equivalence graph satisfying the above properties into a CLASSIC sentence, and vice-versa (Cohen et al., 1992; Borgida & Patel-Schneider, 1992; Cohen & Hirsh, 1994a). Thus, the question of learnability of CLASSIC sentences reduces to that of learning a subclass of equivalence graphs. What would a positive example of an equivalence graph look like? It is another graph which satisfies all of the constraints (and perhaps more) represented by the first. The subsumption algorithm for CLASSIC essentially verifies that the vertex label reached by a path in the first graph is less restrictive than the label of the corresponding vertex (which must exist) in the second graph, and that if in the first graph the two paths labeled by strings w_1 and w_2 lead to the same vertex, then this occurs in the second graph as well. For example, if we add new edges and/or vertices to the graph in Figure 2, we obtain a positive example of the original graph, because it satisfies all of the original constraints, and more. Similarly, if we delete some edge, it becomes a negative example of the original graph. It turns out that the hard part of the learning problem is to determine the structure of the graph and the edge labels, not to determine the vertex labels. Thus, most of the constructors from the CLASSIC language are not problematic; the main challenge is presented by the SAME-AS conditions (each represented by a distinct pair of paths between two vertices), and the role and attributes (which are edge labels). Initially we assume that all the vertex labels are irrelevant; in Section 3 we show how the algorithm is modified when this is not the case.

A natural first attempt to learn such graphs would be to simply intersect the graphs which represent positive examples, thereby extracting the set of common vertices and edges. This approach does not work, as positive examples need not contain all of the edges of the target graph. For example, consider the "universal" positive example graph (Figure 3) consisting of a single vertex, and for each possible edge label, a self-looping edge with that label. For every possible string w, there is a path in this graph labeled with w, and further, for every pair of strings w_1 and w_2, the vertex reached by both is the same (unique) vertex. Hence, *every possible* constraint is satisfied, and this example is a positive example of every

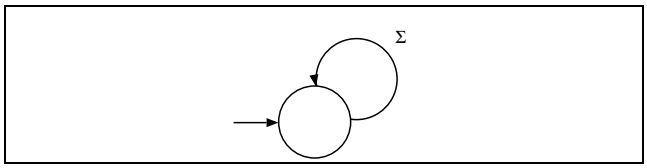

Figure 3. The universally positive example for equivalence graphs. Σ indicates that every possible edge label $\sigma \in \Sigma$ appears on a directed edge from the root vertex back to itself.

possible target graph. But the structure of the target is hidden; the target graph is *not* a subgraph of this positive example graph. Thus, simply intersecting the graphs will not be enough. What does succeed is a variant of the "cross-product of DFAs" construction for regular language intersection. This cross-product will produce a graph corresponding to a CLASSIC description which is maximally specific, and which covers both of the positive examples whose cross-product was taken. By repeatedly taking the cross-product of positive examples, a (one-sided) learning algorithm is obtained. The inadequacy of this approach is easily demonstrated – the cross-product of two equivalence graphs can be as large as the product of their sizes, so the repeated cross-product necessary to implement this approach may yield exponentially sized hypotheses.

Cohen and Hirsh (1994a) circumvent this problem by restricting the number of distinct paths through the graphical representation of a CLASSIC concept. Given a constant k they consider graphs G having at most $|G|^k$ distinct paths (hence their graphs are acyclic). Denote this class k-CLASSIC. They show that the intersection approach above yields an $O(m^{k+1})$ mistake-bounded one-sided learning algorithm for k-CLASSIC, assuming all counterexamples have size at most m. Negatively, they show that in the PAC learning model, assuming that RP $\neq$ NP, CLASSIC is not learnable from random examples alone, even if either of the following constraints hold: (i) the primitive class alphabet is singleton, the role alphabet is doubleton, and the equivalence graph of every example is acyclic, or (ii) the primitive class alphabet is singleton, and the equivalence graph of every example contains only two vertices.

2. Learning Unlabeled Equivalence Graphs

As discussed above, the learning problem for CLASSIC sentences is closely related to that of learning labeled equivalence graphs. We first consider the learning of equivalence graphs without vertex labels, and then indicate how the algorithm is modified to the more general case in Section 3. Later, in Section 4, we modify the algorithm again in order to learn CLASSIC.

Definition 1 *Let Σ be a finite alphabet. A rooted, directed, edge-labeled graph G is an* equivalence graph over Σ *if each vertex v in G is reachable from the root and for every symbol σ in Σ, v has at most one outgoing edge labeled σ. The* size $|G|$ *of an equivalence graph is the sum of the number of edges and vertices in G.*

For the moment, this definition does not recognize any semantic content of the edge labels. However, by representing functions and relations, these edge labels will acquire semantic content when we use equivalence graphs to learn CLASSIC. At that point this definition is to be taken to include the condition that the label of each edge appearing in any pair of edge-disjoint paths between a pair of vertices represents a function.

Simply stated, our algorithm employs the one-sided approach of graph cross-product discussed above, but uses membership queries to bound the intermediate hypothesis size. Figures 4 and 5 give the learning algorithm.

The cross-product $G \times H$ of labeled equivalence graphs G and H is described below, as is the argument that the algorithm efficiently learns. At first glance, equivalence graphs seem DFA-like, but their semantics are quite different, so well-known DFA learning algorithms (Angluin, 1987; Rivest & Schapire, 1993) do not apply. While the algorithm is quite simple, the proof is somewhat subtle. The technical details follow.

Learn

1 Let H be the universally positive example
2 H := **Prune**(H)
3 While **EQUIVALENT**(H) provides counterexample G
4 G := **Prune**(G)
5 H := **Prune**($G \times H$)
6 Return H

Figure 4. Equivalence graphs learning algorithm.

Prune(G)

G is a positive example.
1 For each edge e in G
2 If **MEMBER**($G \backslash e$)[2] is "yes", then remove e from G
3 Return G.

Figure 5. Algorithm using membership queries to remove excess graph elements from a positive example.

A string w of Σ^* is *G-supported* if w is the concatenation of symbols on the edges of a rooted, directed path in G. G defines an equivalence relation $\equiv_G$ on strings of Σ^* as follows: $w_1 \equiv_G w_2$ iff both w_1 and w_2 are G-supported, and their paths terminate at the same vertex. Thus, a G-unsupported string is not G-equivalent to any other string. The set of all strings G-equivalent to a string w is denoted $[w]_G$, and, by an abuse of notation, the set of all strings that terminate at a vertex v of G is denoted $[v]_G$. It is easily verified that for any equivalence graph G, $\equiv_G$ is a right-invariant equivalence relation on strings,

and that if w is G-supported then so is every prefix of w. We now define a partial order on equivalence graphs based on which strings are supported and which strings are equivalent:

Definition 2 *G_1* subsumes *G_2 if every G_1-supported string is G_2-supported and every pair of G_1-equivalent strings are G_2-equivalent.*

It is known (Cohen & Hirsh, 1994a) that a CLASSIC description subsumes a second CLASSIC description iff the equivalence graph of the first subsumes that of the second. Examples will be labeled according to their relationship to the target under this partial ordering. Positive examples of an equivalence graph G will be exactly those graphs G' which are subsumed by G.

Definition 3 *(Cohen & Hirsh, 1994a) Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two equivalence graphs. The* cross-product *of $G_1$and G_2, denoted $G_1 \times G_2$, is defined as follows. Let $p_1, p_2, \ldots, p_{|V_1|}$ denote the vertices of G_1 with p_1 denoting the root, and let $q_1, q_2, \ldots, q_{|V_2|}$ denote the vertices of G_2 with q_1 denoting the root. If G_1 or G_2 is empty, then $G_1 \times G_2$ is empty. Otherwise, the vertex set of $G_1 \times G_2$ (a subset of $V_1 \times V_2$) and the edge set of $G_1 \times G_2$ are defined recursively:*

- *The graph $G_1 \times G_2$ has a root denoted (p_1, q_1).*
- *The graph $G_1 \times G_2$ has a vertex denoted (p_{i_2}, q_{j_2}) and edge $(p_{i_1}, q_{j_1}) \xrightarrow{\sigma} (p_{i_2}, q_{j_2})$ iff $G_1 \times G_2$ has the vertex denoted (p_{i_1}, q_{j_1}), G_1 has edge $p_{i_1} \xrightarrow{\sigma} p_{i_2}$, and G_2 has edge $q_{j_1} \xrightarrow{\sigma} q_{j_2}$.*

Note that $G_1 \times G_2$ is an equivalence graph whenever both G_1 and G_2 are equivalence graphs. The following properties of $G_1 \times G_2$ are either easily verified, or follow from Cohen et al. (1992) and Cohen & Hirsh (1994a).

Property 1 *Let G_1 and G_2 be two equivalence graphs. Then*

1. *A string w is ($G_1 \times G_2$)-supported iff w is both G_1-supported and G_2-supported.*
2. *For any strings s and t, $s \equiv_{G_1 \times G_2} t$ iff both $s \equiv_{G_1} t$ and $s \equiv_{G_2} t$.*
3. *$G_1 \times G_2$ is the most specific generalization (least upper bound) of G_1 and G_2 with respect to the subsumption ordering. That is, if G_1, G_2, and G are equivalence graphs, then if G subsumes both G_1 and G_2, then G subsumes $G_1 \times G_2$.*

Definition 4 *An equivalence graph G is* pruned with respect to *an equivalence graph G_* if G_* subsumes G, but does not subsume any proper subgraph of G.*

The following is a useful property of pruned graphs.

Property 2 *Let G and G_* be two equivalence graphs such that G is pruned with respect to G_*. Then for every vertex v in G and every outgoing edge label σ from v, $[v]_G$ contains some (G_*-supported) string s such that $s\sigma$ is G_*-supported.*

Proof: Suppose to the contrary that G contains a vertex v with outgoing edge label σ such that $[v]_G$ contains no string s such that $s\sigma$ is G_*-supported. But then deleting v's outgoing edge labeled σ produces a proper subgraph of G that supports every G_*-supported string and leaves equivalent every pair of G_*-equivalent strings. This contradicts the hypothesis that G is pruned. ■

LEMMA 1 *Let G_1, G_2, and G_* be equivalence graphs such that both G_1 and G_2 are pruned with respect to G_*. If there exists a vertex v of $G_1 \times G_2$ such that $[v]_{G_1 \times G_2}$ contains only G_*-unsupported strings, then there are two G_*-supported strings that are G_1-equivalent but not ($G_1 \times G_2$)-equivalent.*

Proof: First observe that if G_* supports no strings at all – not even the empty string – then G_*, G_1, and G_2 must each be the empty graph. In this case the lemma holds trivially. The rest of the proof assumes that at least the empty string is supported by G_* and therefore also by G_1 and G_2 by Property 1 item 1. The proof assumes the existence of a vertex v of $G_1 \times G_2$ such that $[v]_{G_1 \times G_2}$ contains only G_*-unsupported strings and then constructs two G_*-supported strings s and s' that are G_1-equivalent but ($G_1 \times G_2$)-inequivalent.

Let v be a vertex of $G_1 \times G_2$ such that $[v]_{G_1 \times G_2}$ contains only G_*-unsupported strings, and let w be any string in $[v]_{G_1 \times G_2}$. Now, since the $G_1 \times G_2$ equivalence class containing the empty string contains a G_*-supported string (the empty string itself) and since no G_*-supported strings are contained in $[w]_{G_1 \times G_2}$, there exists a prefix w_p of w and an edge label σ such that

- $w_p\sigma$ is a prefix of w,
- $[w_p]_{G_1 \times G_2}$ contains a G_*-supported string, and
- $[w_p\sigma]_{G_1 \times G_2}$ contains no G_*-supported string.

Let s be any G_*-supported string in $[w_p]_{G_1 \times G_2}$. Now observe that since w is ($G_1 \times G_2$)-supported so are both $w_p\sigma$ and w_p. Also observe that since $w_p\sigma$ is ($G_1 \times G_2$)-supported $w_p\sigma$ must be G_1-supported by Property 1(1). But by Property 2 since G_1 is pruned with respect to G_*, $[w_p]_{G_1}$ must contain a G_*-supported string s' such that $s'\sigma$ is also G_*-supported. Thus we have two G_*-supported strings s and s' such that $s \equiv_{G_1 \times G_2} w_p$ (which by Property 1(2) implies that $s \equiv_{G_1} w_p$) and $s' \equiv_{G_1} w_p$, and so $s \equiv_{G_1} s'$. Since $s'\sigma$ is G_*-supported, if $s \equiv_{G_1 \times G_2} s'$ then $s' \equiv_{G_1 \times G_2} w_p$ so that $[w_p\sigma]_{G_1 \times G_2}$ contains $s'\sigma$, a G_*-supported string, contradicting the choice of w_p and σ. Thus, $s \not\equiv_{G_1 \times G_2} s'$, and s and s' are G_*-supported strings that are G_1-equivalent but not ($G_1 \times G_2$)-equivalent. ■

The proof that the learning algorithm is correct and efficient (Theorem 1) will follow easily from Lemma 2, which asserts that progress is made with each new hypothesis of the algorithm. The proof of the lemma follows the proof of Theorem 1. We first need the following definition:

Definition 5 *Let G be an equivalence graph that G_* subsumes. Then $\equiv_G^{G_*}$ is the equivalence relation $\equiv_G$ restricted to G_*-supported strings.*

LEMMA 2 *Let G = **Prune**$(G_1 \times G_2)$, with G_1 and G_2 both pruned with respect to G_*. Further suppose that G_1 does not subsume G_2. Then $\equiv_{G^*}^{G_*}$ is a proper refinement of $\equiv_{G_1^*}^{G_*}$.*

THEOREM 1 *Let G_* be the target equivalence graph to be learned. The algorithm* **Learn** *finds an equivalence graph equivalent to G_*, and at no point during execution does the running time exceed a polynomial in $|\Sigma|$, $|G_*|$, and the size of the largest counterexample seen so far.*

Proof: The initial hypothesis has one equivalence class. A simple inductive proof shows that each time **EQUIVALENT**(H) is called, H is a positive example, so by Lemma 2, each counterexample causes the number of equivalence classes over supported strings to increase by at least one. Thus, to show that the number of equivalence queries is bounded above by $|G_*|$, it suffices to show that the number of equivalence classes in the hypothesis is bounded above by the number of equivalence classes of G_*. To see this, note that if some vertex of the hypothesis contained no G_*-supported string, that vertex would have been pruned. On the other hand, if the number of vertices in the hypothesis containing G_*-supported strings is greater than the number of vertices in G_*, then the hypothesis is a negative example because some pair of G_*-equivalent strings are inequivalent in the hypothesis.

To bound the running time of the algorithm, we show that at each step, if $\tilde{G}$ is the counterexample with the greatest number of vertices seen so far, the algorithm has made at most $|G_*|^2 \cdot |\tilde{G}| \cdot |\Sigma|$ membership queries, and has run for at most a number of steps that is polynomial in $|G_*|$, $|\tilde{G}|$, and $|\Sigma|$. This follows from the fact that at each step, if $\tilde{G}$ is the counterexample having the greatest number of vertices seen so far, the number of membership queries used by **Prune** on $H \times \tilde{G}$ is at most $O(|\Sigma| \cdot n_H \cdot n_{\tilde{G}})$, where n_H and $n_{\tilde{G}}$ are the number of vertices in H and $\tilde{G}$, respectively. Since n_H is bounded above by n_{G_*} and since $|H| < |G_*|$ (for every edge of H that could not be deleted by **Prune**, there is some equivalence class, i.e., vertex, of G_* to which that edge can be associated — thus, the number of edges of H is at most the number of edges in G_*, so $|H| < |G_*|$) the number of membership queries used by a single call to **Prune** is $O(|\Sigma| \cdot |G_*| \cdot |\tilde{G}|)$, where $\tilde{G}$ is the largest counterexample yet witnessed by the algorithm. ■

Proof of Lemma 2: It is sufficient to show that $\equiv_{G_1 \times G_2}^{G_*}$ is a proper refinement of $\equiv_{G_1}^{G_*}$ because $\equiv_{G^*}^{G_*}$ is a proper refinement of $\equiv_{G_1 \times G_2}^{G_*}$ (noting that G is obtained by pruning from $G_1 \times G_2$, and no edges or vertices are added — only deleted.)

Now, $\equiv_{G_1 \times G_2}$ is a refinement of $\equiv_{G_1}$ (Property 1(2)). Further, since both are subsumed by G_*, they both support every G_*-supported string. Hence, $\equiv_{G_1 \times G_2}^{G_*}$ is a refinement of $\equiv_{G_1}^{G_*}$. If the number of equivalence classes of $\equiv_{G_1 \times G_2}^{G_*}$ exceeds the number of equivalence classes of $\equiv_{G_1}^{G_*}$, then the lemma is proved. Otherwise, since $\equiv_{G_1 \times G_2}^{G_*}$ is a refinement of $\equiv_{G_1}^{G_*}$, the number of equivalence classes must be the same, and the classes must be identical. We show that this leads to a contradiction, thus proving the lemma.

By Property 1(2), for any strings x and y, $x \equiv_{G_1 \times G_2} y$ iff $x \equiv_{G_1} y$ and $x \equiv_{G_2} y$, and since all three support all G_*-supported strings, we have that for any G_*-supported strings

x and y, $x \equiv^{G_*}_{G_1 \times G_2} y$ iff $x \equiv^{G_*}_{G_1} y$ and $x \equiv^{G_*}_{G_2} y$. This, together with our assumption that the relations $\equiv^{G_*}_{G_1 \times G_2}$ and $\equiv^{G_*}_{G_1}$ are identical, implies that the relations $\equiv^{G_*}_{G_1}$ and $\equiv^{G_*}_{G_2}$ are identical.

By the hypothesis of this lemma, G_1 does not subsume G_2, hence there exist strings t_1 and t_2 such that $t_1 \equiv_{G_1} t_2$, but $t_1 \not\equiv_{G_2} t_2$. (Otherwise, G_1 supports some t that G_2 does not; but then t is G_1-equivalent to some G_*-supported w (by Property 2, since G_1 is pruned). But t is not G_2-equivalent to w, since t is not supported in G_2. In this case, let $t_1 = t$ and $t_2 = w$.)

Clearly, both t_1 and t_2 are supported in G_1.

Case 1: both t_1 and t_2 are supported in G_2. Since both t_1 and t_2 are supported in both G_1 and G_2, they are both supported in $G_1 \times G_2$. Since they are not equivalent in G_2, they are not equivalent in $G_1 \times G_2$. Let v be the vertex that t_1 and t_2 both go to in G_1, and let v_1 and v_2 ($v_1 \neq v_2$) be the vertices that t_1 and t_2 go to, respectively, in $G_1 \times G_2$. Since G_1 is pruned with respect to G_*, there exists a G_*-supported string w that goes to v in G_1 (Property 2). If there do not exist G_*-supported strings w_1 and w_2 such that w_1 and w_2 go to v_1 and v_2 in $G_1 \times G_2$, respectively, then Lemma 1 applies, and we conclude that G_1 and $G_1 \times G_2$ are not equivalent on G_*-supported strings, a contradiction. Since t_1 and w_1 are equivalent in $G_1 \times G_2$, and since t_2 and w_2 are equivalent in $G_1 \times G_2$, we must have the same equivalences in G_1 (noting that G_1 supports all four strings). But t_1 and t_2 are equivalent in G_1, hence w_1 and w_2 are equivalent in G_1, transitively. But w_1 and w_2 are not equivalent in $G_1 \times G_2$, contradicting our assumption that $G_1 \times G_2$ and G_1 define the same relation on G_*-supported strings.

Case 2: at least one of t_1 and t_2 is not supported in G_2. Without loss of generality, assume t_1 is not supported by G_2. Let $t\sigma$ be the shortest prefix of t_1 such that t is G_2-supported, but $t\sigma$ is not G_2-supported. (Such a prefix exists, since the empty string is supported.)

Both G_1 and G_2 support t, so consider the two paths that t induces in these two graphs; we claim that the $\equiv^{G_*}_{G_1}$ equivalence class containing t is the same as the $\equiv^{G_*}_{G_2}$ equivalence class containing t. Suppose by way of contradiction that this is not the case. Now $t = \sigma_1\sigma_2\cdots\sigma_{|t|}$, so look at the first i such that $\sigma_1\sigma_2\cdots\sigma_i$ is contained in non-identical equivalence classes of $\equiv^{G_*}_{G_1}$ and $\equiv^{G_*}_{G_2}$. Since G_1 and G_2 are pruned, the equivalence class containing $\sigma_1\sigma_2\cdots\sigma_{i-1}$ must contain some G_*-supported (possibly empty) string w such that $w\sigma_i$ is also G_*-supported. Now since the equivalence classes of $\equiv^{G_*}_{G_1}$ and $\equiv^{G_*}_{G_2}$ are identical, the $\equiv^{G_*}_{G_1}$ and $\equiv^{G_*}_{G_2}$ equivalence classes containing $w\sigma_i$ must be identical by right invariance. But $w\sigma_i$ is both G_1- and G_2-equivalent to $\sigma_1\sigma_2\cdots\sigma_i$, contradicting the assumption that σ_i is the first i where $\sigma_1\sigma_2\cdots\sigma_{|t|}$ reaches non-identical equivalence classes in $\equiv^{G_*}_{G_1}$ and $\equiv^{G_*}_{G_2}$. Hence t is in identical $\equiv^{G_*}_{G_1}$ and $\equiv^{G_*}_{G_2}$ equivalence classes, completing the proof of the claim. Now, since $[t]_{G_2}$ contained no outgoing edge labeled σ, for no G_*-supported string w in $[t]_{G_2}$ is $w\sigma$ G_*-supported. But then the outgoing edge from $[t]_{G_1}$ labeled σ can be deleted from G_1, contradicting our assumption that G_1 was pruned with respect to G_*, and completing the proof of Lemma 2. ■

3. Learning Labeled Equivalence Graphs

We consider extending the class of equivalence graphs to allow for labeled vertices. The set of vertex labels is required to possess enough structure to allow computing what will be the unique most specific generalization (least upper bound) between any pair of vertex labels. Specifically, the structure we require is a lattice of finite depth. The following definition supplies the notion we adopt.

Definition 6 *Let Σ be an alphabet, and let $\mathcal{L} = \langle \Gamma, \bot, \preceq, \sqcup \rangle$ be a lattice of finite depth (i.e., no infinite chains), with partial order $\preceq$ over elements of the set Γ, having (unique) minimum element $\bot$, and having the binary join operator $\sqcup$ that returns the unique least upper bound of its two operands. Then an $\mathcal{L}$-*labeled equivalence graph *over Σ is a graph G that is an equivalence graph over Σ in which each vertex v of G has been labeled with some $\gamma \in \Gamma$.*

Observation:. If $\{\mathcal{L}_i\}_{i=1}^m$ is a collection of lattices of finite depths $d_1, d_2, \ldots, d_m$, respectively, then the tuples $\{\langle \gamma_1, \ldots, \gamma_m \rangle : \gamma_i \in \Gamma_i\}$ form a lattice of depth $\sum_{i=1}^m d_i$, with minimum element $\langle \bot_1, \ldots, \bot_m \rangle$, and with partial ordering defined by $\langle \gamma_1, \ldots, \gamma_m \rangle \preceq \langle \gamma'_1, \ldots, \gamma'_m \rangle$ exactly when $\gamma_i \preceq_i \gamma'_i$ for each i. We will use this observation later when we return to our discussion of CLASSIC.

For any labeled equivalence graph G and any string w that is G-supported, let $\ell_G(w)$ denote the label of the vertex reached by w. The earlier subsumption-induced partial order for equivalence graphs is modified to account also for vertex labels:

Definition 7 *Let G_1 and G_2 be two $\mathcal{L}$-labeled equivalence graphs. Then G_1* subsumes *G_2 if the conditions of Definition 2 hold, and if in addition, $\ell_{G_2}(w) \preceq \ell_{G_1}(w)$ for every G_1-supported string w.*

It is shown in (Cohen & Hirsh, 1994a) that a CLASSIC description subsumes a second CLASSIC description iff the labeled equivalence graph of the first subsumes that of the second. Thus, positive examples of a labeled equivalence graph G will be graphs G' which are subsumed by G.

The cross-product operation on equivalence graphs given in Definition 3 is easily modified to incorporate vertex labels from a partial order (Cohen & Hirsh, 1994a). The cross-product of two labeled equivalence graphs is just as in Definition 3, but in addition, the label of any vertex (p_i, q_i) that appears in $G_1 \times G_2$ is just the label $\ell_{G_1}(p_i) \sqcup \ell_{G_2}(q_i)$.

Again, the central challenge for learning these graphs is in discovering the structure of the graph, not in determining the vertex labels. In the previous section we presented an algorithm for learning the structure of a graph assuming the vertex labels were unimportant. It is a simple matter to fold the computation of the vertex labels into computing the $G_1 \times G_2$ from G_1 and G_2 – the vertex label for vertex v of $G_1 \times G_2$ is the least upper bound of the label of vertex v_1 of G_1 and the label of vertex v_2 of G_2, where v_1 and v_2 were the vertices that "combined" to produce v.

THEOREM 2 *Let $\perp$ be the minimum element of a lattice $\mathcal{L}$ having maximum chain length d and having polynomial time computable join operator $\sqcup$, and let Σ be a set of edge labels. Then the algorithm composed of figures 4 and 5 learns $\mathcal{L}$-labeled equivalence graphs over Σ from membership and equivalence queries in time polynomial in $|\Sigma|$, d, the longest counterexample received, and the size of the target concept.*

Proof: The algorithm is modified only in that the sole vertex of the initial hypothesis is labeled by $\perp$ and the cross-product operator for labeled equivalence graphs is used; specifically, **Prune** does not change – only edge deletions are attempted.

Observe that a counterexample G to H must be of one of the following (not necessarily mutually exclusive) types

- Some string w supported by both H and G has $\ell_H(w) \prec \ell_G(w)$,
- Some H-supported string is not G-supported, or
- Some pair of H-equivalent strings are G-inequivalent.

If the counterexample G was of either of the latter two types, then G is a counterexample to H based solely on their underlying (non-vertex-labeled) equivalence graphs. Therefore, Lemma 2 applies to show that $\equiv^{G_*}_{\mathbf{Prune}(H \times G)}$ is a proper refinement of $\equiv^{G_*}_{H}$. This can happen at most as many times as there are vertices in G_*.

If the counterexample is only of the first type, then the underlying (non-vertex-labeled) equivalence graphs of H and G (and therefore, **Prune**$(H \times G)$) are isomorphic. As such, some vertex label of H was generalized. Thus, this first type of counterexample can happen at most $dn_H < d \cdot |H| < d \cdot |G_*|$ times between occurrences of a counterexample of the second or third type, where n_H is the number of vertices in H.

A naïve analysis assumes that in the worst case the label on every vertex must be updated d times between changes to the equivalence classes of the hypothesis. This produces a bound of $O(dn^2_{G_*})$ on the number of counterexamples received, where n_{G_*} is the number of vertices in G_*.

A more careful analysis recognizes that, until a collection of target equivalence classes were split, generalizing the vertex label for one of the equivalence classes generalized the label for *every* target equivalence class in the collection; thus every target equivalence class need only be isolated once and have its vertex label changed at most d times overall. This bounds the number of counterexamples by $O(dn_{G_*})$. Thus the algorithm witnesses $O(d \cdot |G_*|)$ counterexamples to its equivalence queries.

As in the case of (non-vertex-labeled) equivalence graphs, the number of membership queries used by **Prune** on $H \times G$ is at most $O(|\Sigma| \cdot n_H \cdot n_G)$, where n_H and n_G are the number of vertices in H and G, respectively, so that the number of membership queries used by a single call to **Prune** is $O(|\Sigma| \cdot |G_*| \cdot |\tilde{G}|)$, where $\tilde{G}$ is the largest counterexample yet witnessed by the algorithm.

Finally, since $\sqcup$ can be applied in polynomial time, and since $|H| < |G_*|$ for each hypothesis H, the class of labeled equivalence graphs are polynomial time learnable using membership and equivalence queries. ■

4. Application to CLASSIC

The graphical representation of CLASSIC (Borgida & Patel-Schneider, 1992; Cohen et al., 1992; Cohen & Hirsh, 1994a) forms the structure of the graph from the AND, ALL, and SAME-AS constraints and annotates the vertices of the graph with the AT-LEAST, AT-MOST, FILLS, ONE-OF, and PRIM constraints. These latter constraints naturally correspond to the vertex labels discussed in the previous section. Moreover, these different kinds of annotating constraints also combine naturally into tuples that can serve as the actual vertex label lattice $\mathcal{L}$, because ordering tuples $\langle s_1, \ldots, s_k \rangle \preceq \langle t_1, \ldots, t_k \rangle$ exactly when every $s_i \preceq t_i$ is in accordance with the notion of subsumption for CLASSIC (Cohen et al., 1992).

We would like to exploit this similarity to labeled equivalence graphs by employing a prediction preserving reduction (Angluin & Kharitonov, 1995, Pitt & Warmuth, 1990) using the labeled equivalence graph learning algorithm developed in the previous section. The reduction would use known polynomial time transformations between CLASSIC descriptions and their graphical representations to turn the CLASSIC description examples into a form suitable for **Learn** and to turn the graphical representations queried and hypothesized by **Learn** into CLASSIC descriptions suitable for examination outside of **Learn**. Unfortunately, the semantics imposed on the graphical representation of CLASSIC dissuade us from this black box approach; we will employ the known transformations only after modifying **Learn**.

4.1. Dealing with mismatches between CLASSIC and equivalence graphs

As cautioned earlier, when we use equivalence graphs to learn CLASSIC, the labels appearing in the graphs acquire semantic content requiring us to restrict to attributes those labels appearing on edge-disjoint paths between any pair of vertices. In order to apply the equivalence-graph learning algorithm to the problem of learning CLASSIC descriptions, we must ensure that every hypothesis entertained by the algorithm corresponds to a valid CLASSIC description. In particular, the universal positive example, used initially by **Learn**, does not satisfy the edge label constraint. In lieu of a universally positive example that satisfies every constraint of the target, we rely on the semantics of the AT-LEAST and AT-MOST constructors to build a CLASSIC description that always denotes the empty set, which is guaranteed to be a subset (and therefore a positive example) of any target description. Concretely, let r be any role. Then

(AND
(AT-LEAST 1 r)
(AT-MOST 0 r))

always denotes the empty set; such a concept is said to be *inconsistent*. (A number of description logics, including more recent versions of CLASSIC, include special descriptions NOTHING and EVERYTHING, to denote the empty set, and the set of all individuals, respectively). Making an equivalence query on the graph of this concept will provide the learner with a positive counterexample, G_0, that satisfies all the constraints of the target. The graph G_0 serves the purpose of the initial universal positive example H used as the

initial hypothesis by **Learn**. A simple inductive proof shows that all subsequent hypotheses of **Learn** are in fact labeled equivalence graphs whose edge labels adhere to this restriction on equivalent strings over the role alphabet.

Because the AND, ALL, and SAME-AS constraints of the target CLASSIC sentence determine only the structure of the target equivalence graph (and are independent of the vertex labels), Theorem 1 applies directly to show that CLASSIC sentences containing only AND, SAME-AS, and ALL statements can be efficiently learned with equivalence and membership queries. In order to apply Theorem 2 to learn arbitrary CLASSIC sentences, we need only argue that the vertex labels arising from the CLASSIC statements {PRIM, AT-LEAST, AT-MOST, FILLS, ONE-OF} has semantics consistent with Definition 6. More specifically, we'll show that the set of vertex labels to be learned at any vertex is an element of a finite-depth lattice whose join operation exactly captures the semantics of subsumption for the label type. In fact, due to our earlier *Observation* regarding the melding of several lattices into a single one, we may treat each of these vertex-label types separately. We will show that in each case, the label sought is either an element of a finite-depth lattice, or else we can apply standard techniques from learning theory to bound the infinite chains of the lattice to only a finite-depth sublattice which contains the label to be learned (and the learning algorithm can easily identify the sublattice to be searched). We proceed with the details.

PRIM: The PRIM statements have the most straightforward structure: Consider the labeled equivalence graph G associated with a given CLASSIC sentence S, and consider a given vertex v. For any particular primitive symbol p, either v contains the constraint (PRIM p), or it does not. Let $\langle$`no-constraint`$\rangle$ denote the absence of such a PRIM constraint for p. Then (PRIM p) $\preceq \langle$`no-constraint`$\rangle$, and each vertex can be thought of as having a label taken from this depth-two chain with minimum element (PRIM p). Intuitively, this just says nothing more than the following: If $v_1 \in G_1$ and $v_2 \in G_2$, then the vertex $\langle v_1, v_2 \rangle$ in the cross-product will have label (PRIM p) if and only if both v_1 and v_2 do; otherwise it will have "label" $\langle$`no-constraint`$\rangle$.

It should be noted that although CLASSIC may allow for an infinite number of different primitive symbols, the learning algorithm need only ever concern itself with those primitive symbols p for which the constraint (PRIM p) actually appears in some vertex of the first counterexample G_0. (Call this set of primitive symbols $\mathcal{P}_0$.) By the semantics of PRIM and subsumption, any primitive not in $\mathcal{P}_0$ cannot appear in the target G_*. Thus, the PRIM constraint results in $d_0 = |\mathcal{P}_0|$ independent depth-2 chains that will get combined as described in the *Observation*.

AT-LEAST: For any non-negative integers n and m, and for any role r, (AT-LEAST n r) $\preceq$ (AT-LEAST m r) if and only if $n \geq m$. Consequently, there is no minimum (least general) such constraint. However, the first counterexample G_0 obtained is a positive example that does not denote the empty set, and any AT-LEAST constraint appearing in G_0 finitely bounds the AT-LEAST lattice as follows. Let $d_1 = \max\{x :$ for some role r and vertex v in G_0, vertex v has the constraint (AT-LEAST x r)$\}$. Then the minimum (least general) element for any vertex and any role appearing in the target G_* can be assumed to be no more specific than (AT-LEAST d_1 r). Thus, for each role r the AT-LEAST constraints

are elements of a lattice (in fact, a chain), with join operator given by (AT-LEAST n r) $\sqcup$ (AT-LEAST m r) = (AT-LEAST $\min\{n, m\}$ r). The minimum element is (AT-LEAST d_1 r), and the maximum (most general) element is (AT-LEAST 0 r), which is equivalent to the null constraint $\langle$`no-constraint`$\rangle$. The depth of the chain is $d_1 + 1$.

AT-MOST: For any non-negative integers n and m, and for any role r, (AT-MOST n r) $\preceq$ (AT-MOST m r) if and only if $n \leq m$. Further, for any value of k, (AT-MOST k r) $\preceq$ $\langle$`no-constraint`$\rangle$. Thus, the partial order induced by the semantics of AT-MOST yields a structure corresponding to the ordinal $\omega + 1$: an infinite chain with minimum element (AT-MOST 0 r), together with a single maximum element that corresponds to the absence of any AT-MOST constraint. While we have no need to use G_0 to obtain a minimum element as in the case of AT-LEAST, we have a slightly different problem: The most general AT-MOST constraint is the absence of an AT-MOST constraint, which is not an AT-MOST constraint at all. Consequently, a sequence of (positive) counterexamples of the form (AT-MOST i r), for $i = 1, 2, 3, \ldots$, forces an algorithm doing simple most-specific-generalizations to hypothesize exactly each counterexample (AT-MOST i r) after it is received. No positive example ever provides reason to eliminate the AT-MOST constraint. Thus, if the target G_* had no AT-MOST constraint for a particular vertex and role, then this fact would never be discovered.

To overcome this difficulty, we employ a technique in which the learning algorithm correctly "guesses" the value of some unknown parameter. (We discuss later how to handle this efficiently in a deterministic setting.) In particular, let d_2 be the value of the largest integer appearing in any AT-MOST constraint in any vertex of the target graph G_*, for any role r. Armed with this knowledge, a finite chain is induced for each role r—we may eliminate all constraints of the form (AT-MOST n r) for $n > d_2$, but we keep the maximum element $\langle$`no-constraint`$\rangle$. Thus, the learning algorithm assumes for every role r and every vertex v, that either there is a constraint (AT-MOST n r) at v for n between 0 and d_2, or else there is no such AT-MOST constraint for r. In other words, the constraint is a member of the chain of depth $d_2 + 2$ of the form (AT-MOST 0 r) $\preceq$ (AT-MOST 1 r) $\preceq$ (AT-MOST 2 r) $\preceq \cdots \preceq$ (AT-MOST d_2 r) $\preceq \langle$`no-constraint`$\rangle$, with join operator defined by (AT-MOST n r) $\sqcup$ (AT-MOST m r) = (AT-MOST $\max\{n, m\}$ r) if $n, m \leq d_2$, or = $\langle$`no-constraint`$\rangle$ otherwise. Note that a finite chain is also induced by any upper bound $d_2' > d_2$, so thatif the "guess" of d_2 is too large, the search will still succeed.

FILLS: By definition of the semantics of FILLS, for any role r and disjoint primitives $p_1, p_2, \ldots, p_n$ and $q_1, q_2, \ldots, q_m$, (FILLS r $p_1, p_2, \ldots, p_n$) $\preceq$ (FILLS r $q_1, q_2, \ldots, q_m$) if and only if $\{q_1, q_2, \ldots, q_m\} \subseteq \{p_1, p_2, \ldots, p_n\}$. Moreover, it is easily argued that the most specific generalization of (FILLS r $p_1, p_2, \ldots, p_n$) and (FILLS r $s_1, s_2, \ldots, s_m$) is (FILLS r $t_1, t_2, \ldots, t_k$), where $\{t_1, \ldots, t_k\} = \{p_1, \ldots, p_n\} \cap \{s_1, \ldots, s_m\}$. Consequently, we may again use the initial counterexample G_0 to obtain an initial set of possible disjoint primitives for each role that may appear in the target concept at any vertex. In particular, for each role r, let F_r denote the set of all primitive symbols that appear in any FILLS constraint involving r at any vertex of G_0. Then every FILLS constraint involving r in the target G_* is of the form (FILLS r $q_1, q_2, \ldots, q_n$), where each $q_i \in F_r$. Thus, after the first

counterexample G_0 is obtained, we have a finite lattice for each role r constraint (FILLS $r \ldots$) that appears in the target. The depth is $d_3 + 1$, where $d_3 = |F_r|$. The join operation for the lattice is set intersection, and the minimum element is (FILLS r $\langle empty - list \rangle$) which corresponds to $\langle$`no-constraint`$\rangle$.

ONE-OF: If $p_1, p_2, \ldots, p_n$ and $q_1, q_2, \ldots, q_m$ are disjoint primitives, then, by the semantics of ONE-OF, (ONE-OF $p_1, p_2, \ldots, p_n$) $\preceq$ (ONE-OF $q_1, q_2, \ldots, q_m$) if and only if $\{p_1, p_2, \ldots, p_n\} \subseteq \{q_1, q_2, \ldots, q_m\}$. Moreover, the most specific generalization of (ONE-OF $p_1, p_2, \ldots, p_n$) and (ONE-OF $s_1, s_2, \ldots, s_m$) is easily seen to be (ONE-OF $t_1, t_2, \ldots, t_k$), where $\{t_1, \ldots, t_k\} = \{p_1, \ldots, p_n\} \cup \{s_1, \ldots, s_m\}$. Thus, a lattice is formed with the join operation being that of set union, and we may choose as initial vertex labels those ONE-OF constraints appearing in the first counterexample G_0. However, a problem similar to that arising in the case of AT-MOST occurs in bounding the maximum depth of the lattice. The nonconvergent infinite sequence of hypotheses (ONE-OF p_1), (ONE-OF p_1, p_2), (ONE-OF p_1, p_2, p_3), ..., of **Learn** may be obtained from the corresponding infinite sequence of counterexamples (ONE-OF p_1), (ONE-OF p_2), (ONE-OF p_3), ... Further, each of these is less general than the lack of any constraint $\langle$`no-constraint`$\rangle$.

We deal with the problem in the same way that AT-MOST was handled. In particular, let d_4 be the maximum cardinality of any set of disjoint primitives appearing in a single ONE-OF constraint in any vertex of the target graph G_*, for any role r. Given knowledge of d_4, we may assume that every ONE-OF constraint contains at most d_4 disjoint primitives. Since proper generalization via the join operation (union, in this case) must increase the cardinality of the disjoint primitive symbols appearing in a ONE-OF constraint, and any proper generalization of the constraint (ONE-OF r $p_1, \ldots, p_{d_4}$) yields $\langle$`no-constraint`$\rangle$, it follows that the lattice corresponding to ONE-OF constraints has depth $d_4 + 1$. Note that a finite lattice is also induced by any upper bound $d'_4 > d_4$.

4.2. Putting it all together

We are now ready to apply Theorem 2 to CLASSIC. We actually obtain a pseudo-polynomial time algorithm, which we strengthen below.

THEOREM 3 CLASSIC *is learnable from membership and equivalence queries in time polynomial in s, t, and m, where s is the number of symbols needed to write the target* CLASSIC *description, t is the number of symbols needed to write the longest counterexample description, and m is the largest integer appearing in any AT-LEAST or AT-MOST constraint in the target description or the first counterexample.*

Proof: Let d_1, d_2, d_3, and d_4 be as described in the preceding section, reproduced here for convenience:

d_0 = $|\mathcal{P}_0|$, where $\mathcal{P}_0$ = primitives appearing in G_0.
d_1 = maximum AT-LEAST number appearing in any vertex of G_0.
d_2 = maximum AT-MOST number appearing in any vertex of G_*.
d_3 = $|\mathcal{F}_r|$, $\mathcal{F}_r$ = disjoint primitives appearing in FILLS constraints in G_0.
d_4 = maximum cardinality of any set of disjoint primitives appearing in a single ONE-OF constraint in G_*.

Consider an algorithm that obtains G_0, and by inspection determines $d_0, d_1,$ and d_3. Assume that the algorithm is given the values of d_2 and d_4. Suppose that V is the total number of vertices in G_* and G_0 together, and that R is the total number of role symbols appearing in G_* or G_0. We count the number of distinct vertex-label lattices with which the learning algorithm must cope, together with the depths of those lattices. Note first that each of V vertices of G_0 or target G_* may contribute d_0 lattices of depth 2 corresponding to PRIM constraints. Next, since there are at most VR vertex-role pairs in both G_* and G_0, it is easily seen that there are at most VR lattices of depth $d_1 + 1$ (respectively, of depths $d_2 + 2$, $d_3 + 1$, $d_4 + 1$) arising from the AT-LEAST (respectively, AT-MOST, FILLS, ONE-OF) constraints. It follows from our earlier *Observation* that the vertex labels combine into a single lattice of vertex labels formed from tuples of PRIM, AT-LEAST, AT-MOST, FILLS, and ONE-OF labels, with depth at most $2d_0V + VR(d_1 + d_2 + d_3 + d_4 + 5)$, and with ordering on the tuples in accordance with the CLASSIC subsumption relation, so that Theorem 2 applies. The total time taken is a polynomial in the above expression, which satisfies the statement of Theorem 3 because s is bounded below by d_0 and d_3, t is bounded below by d_4, $s + t$ is $\Theta(VR)$, and m is bounded below by d_1 and d_2.

To complete the proof it remains to show that without loss of generality, the algorithm described above does not need to be given values d_2 and d_4. The trick is a standard one: The algorithm "guesses" an upper bound m for the maximum of d_2 and d_4. Initially, m is set to 1. Under the assumption that m indeed is an upper bound on d_2 and d_4, the algorithm should correctly learn the target concept within a known time-bound (the exact computation of which is left as an exercise for the reader). If this time bound is exceeded, the algorithm begins again, but doubling the value of m. After at most $O(\log \max\{d_2, d_4\})$ restarts, the assumed upper bound will be sufficient and will be at most twice the minimum sufficient value. This "guessing" produces at most a factor of $O(\log m)$ slowdown over having known m from the outset. ■

Theorem 3 proves only a pseudo-polynomial time algorithm for learning CLASSIC, since the time and number of examples depends polynomially on the *value* of the largest integer m appearing in an AT-LEAST or AT-MOST constraint, and not on the length $\log m$ needed to write the number. There are three methods for strengthening the result: We can incorporate into **Prune** a binary search procedure, using membership queries, to minimize the values in the AT-LEAST constraints, and maximize the values in the AT-MOST constraints, while still retaining a positive example. It can be shown that such a procedure results in a fully polynomial-time algorithm for learning CLASSIC with equivalence and membership queries. However, for reasons that will become apparent in Section 8, we would like to avoid asking membership queries involving only changed vertex labels, so we do not present this approach here. As a second alternative, application of clever

techniques (Chen & Maass, 1994)[3] would probably allow the same binary search on the AT-LEAST and AT-MOST constraint values to be performed, but without relying on membership queries. We leave the details as an exercise for the truly motivated reader.

Instead, for simplicity, we relax the model and argue that a fully polynomial time PAC algorithm (with membership queries) exists for learning CLASSIC. Because the resulting algorithm will not make any membership queries other than those already made by **Prune**, we will be able to apply a general lemma in Section 8 which allows a very noise-tolerant variation.

Recall that PAC learning requires that for any example length n, for any arbitrary unknown probability distribution on examples of length at most n of the target concept G_*, and for any $\epsilon, \delta > 0$, the learning algorithm outputs a concept G that, with probability at least $1-\delta$, has error at most ϵ in classifying random examples as G_* does, where the error is measured with respect to the unknown distribution. The time taken by the learning algorithm, and the number of randomly generated labeled examples that it obtains, are required to be bounded by a polynomial in n, $|G_*|$, $\frac{1}{\epsilon}$, and $\frac{1}{\delta}$. We consider the PAC learning model where the algorithm may ask membership queries in addition to receiving randomly generated examples.

Consider the standard transformation (Angluin, 1988b) of **Learn** into a PAC algorithm: Each equivalence query is replaced by random sampling; each hypothesis is tested to see if it is probably approximately correct. If so, the hypothesis is kept and the algorithm terminates. Otherwise, a counterexample is obtained. The bound on the number of possible equivalence query counterexamples for the original **Learn** translates into a bound on the number of random examples needed for its PAC version. This transformation, together with Theorem 3, gives a fully polynomial time PAC algorithm (with membership queries) for learning CLASSIC descriptions *without* AT-LEAST and AT-MOST constraints. Call this algorithm $\mathbf{Learn_p}$.

Now, to learn CLASSIC *with* AT-LEAST and AT-MOST constraints, first run $\mathbf{Learn_p}$ with appropriate parameters, so as to obtain an hypothesis H that with probability at least $1-\delta/2$ has error at most $\epsilon/2$ if we ignore any misclassification error due solely to AT-LEAST or AT-MOST constraints. If H contains k such constraints (k is at most $2 \cdot |\Sigma| \cdot |G_*|$), then take a sample of $\frac{2k}{\epsilon} \ln \frac{2k}{\delta}$ additional examples. For a given AT-LEAST constraint (AT-MOST similar), let m_0 be the minimum number observed in a positive example for that constraint. Now consider the error induced by using (AT-LEAST m_0 r) instead of using (AT-LEAST m_* r), where $m_* \leq m_0$ is the actual number appearing in the corresponding AT-LEAST constraint of the target. The probability that this error exceeds $\epsilon/2k$ is $(1 - \frac{1}{\epsilon/2k})^{\frac{\delta}{2k} \ln \frac{\epsilon}{2k}} \leq \delta/2k$. Summing over all such constraints, we find that the total error attributable to insufficiently general AT-LEAST or AT-MOST constraints is bounded above by $\epsilon/2$ with probability at least $1 - \delta/2$.

These observations, together with the correctness of Theorem 3, prove the following.

THEOREM 4 CLASSIC *is learnable in the PAC model with membership queries in time polynomial in* $|G_*|$, $\frac{1}{\epsilon}$ $\frac{1}{\delta}$, *and* n, *the upper bound on example size.*

5. The Insufficiency of Membership Queries

We show in this section that efficient learnability cannot be achieved solely through membership queries. Coupled with the result of Cohen and Hirsh (1994a) showing that equivalence queries alone are insufficient assuming P $\neq$ NP (or random examples are insufficient in the PAC setting assuming RP $\neq$ NP), this shows that membership and equivalence queries (or random examples, in the PAC model) form a minimal set of queries with which CLASSIC can be efficiently learned.

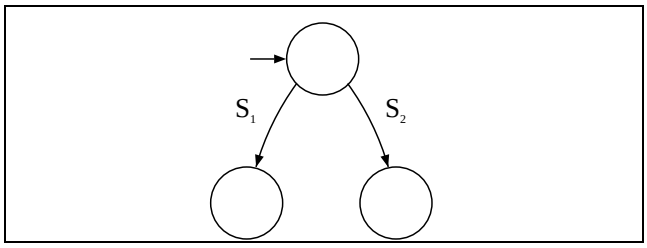

Figure 6. A target schema requiring exponentially many membership queries.

THEOREM 5 *Any algorithm using membership queries alone to learn* CLASSIC *requires*

1. $\Omega(2^{|\Sigma|})$ *membership queries even when the target sentence has an acyclic equivalence graph with only three vertices, and when all roles are attributes, and*
2. $\Omega(2^n)$ *membership queries even when the target sentence has only two roles which are in fact attributes (*$|\Sigma| = 2$*) and when the corresponding equivalence graph is acyclic with* $O(n)$ *vertices.*

Proof: For part (1), consider CLASSIC descriptions whose equivalence graphs have the simple form shown in Figure 6. There are $O(2^{|\Sigma|})$ such concepts, each determined by a partitioning of edge labels[4] of Σ into (disjoint but exhaustive) sets S_1 and S_2. Now, *any* membership query supporting all strings with a single equivalence class is a positive example – no information is obtained by asking such a query. Second, *any* membership query that does not support some edge label from the root is a negative example because a target-supported string is unsupported – membership queries of this form provide no information. Third, *any* membership query that partitions the set of edge labels emanating from the root into more than two sets is a negative example because some pair of target-equivalent strings are inequivalent – membership queries such as these provide no information in distinguishing among the possible targets.

Thus, only membership queries that partition the set of length 1 strings into two supported equivalence classes can provide any information. Any query that does not partition the edge labels into exactly the same sets as the target is a negative example. There are $2^{|\Sigma|-1}$ such partitions. The adversary simply answers any such query "no" until all but one of the partitionings have been exhausted. Notice that if the learner outputs some conjecture

before exhausting all possible partitions, the teacher simply asserts that the conjecture is incorrect by choosing as the target any unexplored partitioning.

For part (2), we simulate a set Σ' of edge labels with $|\Sigma'| = n = 2^k$ by using the labels of Σ as a binary code to label a depth $k-1$ binary tree so that all length $k-1$ strings are supported and every string of length $k-1$ or less is in its own equivalence class. Now construct two more equivalence classes such that every string of length k is in one of these equivalence classes. (See figure 7). These last two equivalence classes simulate the non-root equivalence classes of the target in part (1), so that learning this target requires $2^{n-1}-1$ membership queries, but the target has only $\sum_{i=0}^{k-1} 2^i = O(n)$ vertices. ■

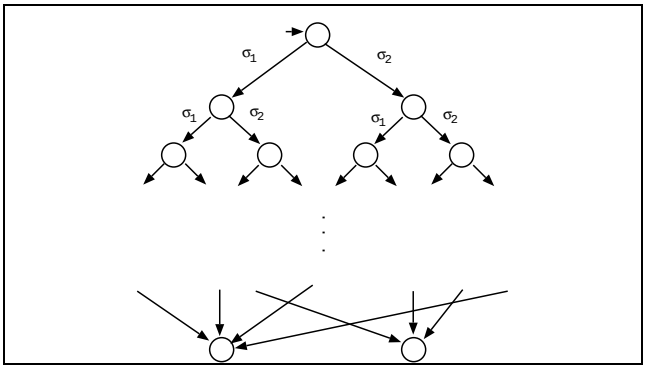

Figure 7. This schema requires exponentially many membership queries even though Σ is known to be the set $\{\sigma_1, \sigma_2\}$.

Having shown that the set of queries we use to achieve our positive learnability result is minimal, the next sections explore extensions to that result.

6. Learning from Individuals

We consider the learning model discussed in Section 1.2, where a relational database K is available that explicitly describes all possible examples. We assume each example individual has a name that is a key, and that the relations of the database are available for inspection by the learning algorithm. Some names are designated as picking out positive examples, and others as negative examples. We are interested in the situation where the database is sufficiently large so as to preclude the strategy of learning by simply collecting positive examples.

In the exact learning model, the goal of the learner is to find a CLASSIC sentence whose denotation on K coincides with exactly the positive examples of K. Any equivalence query on a description H that does not have this property is answered by the name of some individual of K that is classified incorrectly by H. Similarly, in the PAC learning model,

we assume that an arbitrary probability distribution D on (names of) individuals of K is chosen, and the learning algorithm must find, after sampling from D and being told which were positive examples, a description H whose denotation on K has error at most ϵ with probability at least $1 - \delta$.

In an experimental setting, Cohen and Hirsh (1994b) consider a similar type of learning from individuals, although their notion of "individual" may have some additional information attached in the form of a CLASSIC assertion. They adapt a result on learning CLASSIC sentences without the SAME-AS construct, to the context of learning from individuals. The main idea used is to construct, from each individual example, a CLASSIC sentence that abstracts the individual and is as restrictive as possible among all such CLASSIC descriptions. This CLASSIC description is then passed to the learning algorithm as an example. Because the search for the "most restrictive abstraction" of an individual could be combinatorially expensive, they restrict their search to those CLASSIC sentences for which the maximum nesting of ALL constraints is bounded by some constant k. This places a bound on a type of chaining—it embodies the assumption that the data about an individual that is relevant for describing that individual can be obtained by looking at other individuals who are at most some fixed distance k away via relation and function applications.

Indeed, one of the problems in learning from individuals is that via function or relation chaining, each individual can be related to *every* other in the database. The target CLASSIC sentence can specify that an example is a positive example only if some condition holds for individuals that are related by a very long chain of relations from the given example. We exploit just this possibility in proving the negative results in this section, showing that the chaining depth restrictions imposed by Cohen and Hirsh are apparently necessary.

We show that just predicting the classification of CLASSIC sentences on random unseen examples in the PAC setting is as hard as predicting polynomially-sized circuits, and hence is intractable given standard cryptographic assumptions, such as the existence of one-way functions or of cryptographically secure bit generators. (See the work of Angluin & Kharitonov (1995), Kearns & Valiant (1994), Pitt & Warmuth (1990), and Valiant (1984), for nonlearnability results arising from hard cryptographic problems. Moreover, because Boolean circuits are "prediction-complete" for P (Pitt & Warmuth, 1990), if a polynomial-time algorithm existed that could, after seeing a polynomial number of random example individuals of some CLASSIC sentence, labeled as positive or negative, achieve classification accuracy $\frac{1}{2} + \epsilon$ (only slightly greater than a half), then this algorithm could be used (with the appropriate polynomial-time reductions) to achieve accuracy arbitrarily close to 1 for *any* concept class for which the membership problem[5] is decidable in polynomial-time (Pitt & Warmuth, 1990; Schapire, 1990). . Since most reasonable concept learning problems have a polynomial-time membership problem, in some sense then, learning CLASSIC descriptions from individuals is "universal", and is as hard as any reasonable concept learning problem. This negative result holds even when the target CLASSIC description does not contain any SAME-AS conditions.

Whether or not learning from individuals is tractable when membership queries are allowed also, depends on the definition of a membership query in this setting. If such queries simply specify the name of an individual x in the database K, and the response is whether or not x is a positive or negative example of the target description, then our reduction still

holds, and the problem of predicting CLASSIC descriptions from individuals, using random examples and membership queries, is as hard as predicting polynomially sized Boolean circuits with the same information, and is intractable assuming the existence of one-way functions (Angluin & Kharitonov, 1995). On the other hand, if a membership query is allowed to construct a "new individual", one that is not currently in the database, then we leave open the question of whether learning is possible. Which model of membership query on individuals is appropriate, or more natural, depends on our view of whether the database reflects all possible ground instances that are meaningful or relevant.

6.1. Constructing a Database and CLASSIC *Sentence from a Circuit*

We proceed with the technical details. We will show how to encode the behavior of an arbitrary Boolean circuit as a collection of individuals in a database. Being able to predict which entries are positive examples, and which are negative examples will be exactly the problem of determining the input-output behavior of the circuit.

Let B be a Boolean circuit with m gates, where the first n of these gates are input ports. Let the gates be numbered $g_1, g_2, \ldots, g_m$, where the indexing is without loss of generality consistent with a topological sort of the gates. Thus g_m is the output gate, giving the value of the circuit for a given setting of the first n (input) gates, and for each $k > n$, g_k is either an AND or OR gate with inputs g_i and g_j for $i, j < k$, or else it is a NOT gate with input g_i with $i < k$.

Consider the step-by-step process of evaluating B on a given Boolean input vector of length n, as dictated by the topological sort. Initially, $g_1, \ldots, g_n$ are defined, and $g_{n+1}, \ldots g_m$ are undefined. Then, at stage k, g_{n+k} obtains a value determined by the (already-defined) values of the at most two gates g_i, g_j (with $i, j < n + k$) that feed into g_{n+k}.

We can represent the partial evaluation of B by a length m string, where the first n characters take the value 0 or 1, reflecting the original input setting, and the remaining $m - n$ characters take the value 0, 1, or "?", indicating that the value of the corresponding gate has been computed to be either 0, 1, or not yet defined, respectively. There are $2^n 3^{m-n}$ such strings. Each such string will be the name of a unique individual over which the relations of the database are defined. Let this set of domain individuals be denoted I. In what follows, if $s \in I$ and we write $s = xy$; this is to be taken as an indication that x consists of the first n bits of s (corresponding to the input bits), and y is the remaining $m - n$ characters of s (corresponding to the (perhaps partially) evaluated gates). Also, for $s \in I$ and i in the range $\{1, \ldots, m\}$, $s[i]$ denotes the ith character of s.

The database contains the following relations on I.

- INPUT-VECTOR is a unary predicate on I such that INPUT-VECTOR(s) is true if and only if $s = xy$, where $x \in \{0, 1\}^n$, and $y = ?^{m-n}$. Thus, INPUT-VECTOR consists only of those individuals that denote a completely-unevaluated circuit with a particular input vector x.

- EVAL-TO-1 is a unary predicate on I such that EVAL-TO-1(s) is true if and only if $s \in \{0,1\}^m$ and $s[m] = 1$. Thus, s is a completely evaluated circuit whose last bit (i.e., the output bit) is "1".

- For each choice of distinct values i and j in the range $\{1, \ldots, m-1\}$ and for each choice of k in the range $\{\max\{n, i, j\} + 1, \ldots, m\}$, AND$_{i,j,k}$ (respectively, OR$_{i,j,k}$) is a binary relation on I (more specifically, a function from I to I) defined as follows: Let s_1 and s_2 both be elements of I. Then $(s_1, s_2) \in$ AND$_{i,j,k}$ (respectively, OR$_{i,j,k}$) if and only if

  1. $s_1[i]$ and $s_1[j]$ are in {0,1} (i.e., the ith and jth gates of the partially evaluated circuit that s_1 represents are defined (not equal to "?"));
  2. $s_1[k] =$"?" (i.e., the kth gate of the partially evaluated circuit represented by s_1 has not yet been defined);
  3. For every number b in the range $\{1, \ldots, m\}$, if $b = k$ then $s_2[b] = s_1[i] \wedge s_1[j]$ (respectively, $= s_1[i] \vee s_1[j]$), otherwise $s_2[b] = s_1[b]$. Thus, s_2 represents virtually the same partially evaluated circuit as s_1, except that s_2 denotes exactly one further step of evaluation: the values of the gates numbered i and j have been conjoined (respectively, disjoined) to obtain the value of the kth gate.

- For each choice of i in the range $\{1, \ldots, m-1\}$ and for each choice of k such that $i < k \leq m$, NOT$_{i,k}$ is a binary relation on I (more specifically, a function from I to I) defined as follows: Let s_1 and s_2 both be elements of I. Then $(s_1, s_2) \in$ NOT$_{i,k}$ if and only if

  1. $s_1[i] \in \{0,1\}$ (i.e., the ith gate of the partially evaluated circuit that s_1 represents is defined (not equal to "?"));
  2. $s_1[k] =$"?" (i.e., the kth gate of the partially evaluated circuit represented by s_1 has not yet been defined);
  3. For every number b in the range $\{1, \ldots, m\}$, if $b = k$ then $s_2[b] = \neg(s_1[i])$, otherwise $s_2[b] = s_1[b]$. Thus, s_2 represents virtually the same partially evaluated circuit as s_1, except that s_2 denotes exactly one further step of evaluation: the value of the gates numbered i has been negated to obtain the value of the kth gate.

This completes the description of the database K containing all individuals I, and relations INPUT-VECTOR, EVAL-TO-1, AND$_{i,j,k}$, OR$_{i,j,k}$, and NOT$_{i,k}$, for $i, j,$ and k in the specified ranges. As mentioned earlier, the number of individuals is $|I| = 2^n 3^{m-n}$, one entry for each string s as above. Note that each element of one of the relations is either a pair of elements of I (for the binary relations AND, OR, NOT), or a single element of I (for the predicates INPUT-VECTOR and EVAL-TO-1), so that each tuple of any of the relations has a description that is of size polynomial in the size of the circuit.

We now specify, given a Boolean circuit B of m gates (the first n of which are inputs, and the last of which is output), a CLASSIC description whose denotation on K corresponds exactly to the elements $s \in I$ such that

(1) The key s corresponds to an input vector of B, without any partial circuit evaluation completed. That is, $s = xy$, where x is a bit string of length n representing a circuit input, and y is a string of $m - n$ "?"'s.

(2) $B(x) = 1$.

While reading the following description, the reader may find it helpful to refer to Section 6.2, which provides a concrete example of the reduction. The CLASSIC description will classify an entry with key s as a positive example if and only if the two conditions (1) and (2) are satisfied. Clearly, a CLASSIC sentence S_1 asserting that the entry s has the form (1) above is just the sentence S_1 given by

$$S_1 = \text{(INPUT-VECTOR)}.$$

To assert the second condition, consider the step-by-step evaluation of the function computed by B via computing at each step, the output of the next gate g_k in the topological ordering. This is achieved by starting in the database with entry $s = xy$ for which INPUT-VECTOR(s) holds, successively following pointers as given by the function fields $\text{AND}_{i,j,k}$, $\text{OR}_{i,j,k}$, and $\text{NOT}_{i,j}$, to chain through all the $m - n$ entries representing the circuit's partially evaluated values until a final value $s = xy'$ is reached, where y' represents the outputs of all gates if x is the input to B. In particular, the function string $f_m(f_{m-1}(\ldots(f_{n+3}(f_{n+2}(f_{n+1}(s))))))$ will reach the appropriate entry xy' exactly when the function f_k (with $m \geq k \geq n + 1$) is chosen to be $\text{AND}_{i,j,k}$ (respectively, $\text{OR}_{i,j,k}$ or $\text{NOT}_{i,k}$) if gate g_k computes the AND of gates g_i and g_j, (respectively OR of g_i and g_j, or NOT of g_i).

Then $s = xy$ is a positive example if the name xy' of this last individual reached, which represents the completely evaluated circuit, happens to have a 1 in its last position. By construction, this occurs exactly when xy' satisfies the unary predicate EVAL-TO-1(xy').

Consequently, the condition (2) above can be satisfied by the CLASSIC sentence S_2 given by

$$S_2 = \text{(ALL } f_{n+1} \text{ (ALL } f_{n+2} \text{ (ALL } \ldots \text{ (ALL } f_{m-1} \text{ (ALL } f_m \text{ EVAL-TO-1)))))}$$

where each f_i is chosen to correspond, as discussed above, to the function computed by the corresponding gate of B.

Finally, the CLASSIC sentence S that realizes the entire construction is simply

$$S = (\text{AND } S_1 \ S_2).$$

It is immediate from the construction that the denotation of S (the positive examples) on the database K consists exactly of those individuals in I with name of the form xy, where x is an n-bit Boolean vector such that $B(x) = 1$, and where y is a string of $m - n$ "?"'s.

6.2. An Example

Suppose the Boolean circuit B has 8 gates, and computes the function $x_3 \wedge (x_1 \vee x_2) \vee \neg x_2$, as shown in Figure 8. Then individuals will be indexed by strings of length 8, where the

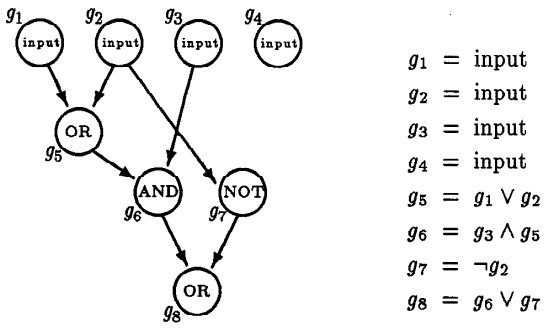

Figure 8. Example circuit and the straight-line program that computes it. Gate g_8 is the output.

first four bits are either 0 or 1, and the last four bits are either 0, 1, or ?. The value of the circuit can be computed by the following function composition:

$$\text{(EVAL-TO-1 (OR}_{6,7,8}\text{ (NOT}_{2,7}\text{ (AND}_{3,5,6}\text{ (OR}_{1,2,5}(x)))))\text{)}$$

For example, to see how this evaluates on input 0111, we'll look at the partial values as the functions are applied from innermost to outermost.

(EVAL-TO-1 ($\text{OR}_{6,7,8}$ ($\text{NOT}_{2,7}$ ($\text{AND}_{3,5,6}$ ($\text{OR}_{1,2,5}$ (0111????))))))
= (EVAL-TO-1 ($\text{OR}_{6,7,8}$ ($\text{NOT}_{2,7}$ ($\text{AND}_{3,5,6}$ (01111???)))))
= (EVAL-TO-1 ($\text{OR}_{6,7,8}$ ($\text{NOT}_{2,7}$ (011111??))))
= (EVAL-TO-1 ($\text{OR}_{6,7,8}$ (0111110?)))
= (EVAL-TO-1 (01111101))
= 1

The last four bits give the values of all of the gates when the input is 0111. Thus, if we consider the individual named 0111????, and chain through the database following the appropriate function fields as given by the sequence

$$\text{OR}_{6,7,8}(\text{NOT}_{2,7}(\text{AND}_{3,5,6}(\text{OR}_{1,2,5}())))$$

then we end up at the individual named 01111101. Because the last bit of this is a 1, EVAL-TO-1(01111101) = 1. Hence 0111???? is a positive example of the CLASSIC sentence

(**AND** (INPUT-VECTOR)
  (**ALL** $\text{OR}_{1,2,5}$ (**ALL** $\text{AND}_{3,5,6}$ (**ALL** $\text{NOT}_{2,7}$ (**ALL** $\text{OR}_{6,7,8}$ (EVAL-TO-1))))))

6.3. Learning from Individuals Implies Circuit Prediction

We have shown that for any circuit B there exists a database K of individuals and relations among individuals, and a CLASSIC sentence S, such that the denotation of S on K is exactly the entries of K that have names that "correspond" to inputs for which B outputs 1.

We use these constructions to sketch how an algorithm that learns CLASSIC descriptions can be used to predict, according to the PAC requirements, the values of a hidden target circuit B given only random examples, and membership queries on input vectors to B.

Let L be a learning-from-individuals algorithm for CLASSIC sentences. Suppose B is an unknown target circuit of n inputs and $m - n$ gates to be predicted. (Without loss of generality, we assume that for the circuit learning problem, we know the number of gates in the target.) We describe a learning algorithm L' that uses L for predicting B's behavior.

A minor problem which does not appear with standard "learning reductions" is that the learning algorithm L assumes the existence of an exponentially-sized database from which examples are drawn, and from which it can extract information. The (efficient) reduction that we now describe cannot afford to entirely construct the exponentially large database K corresponding to B. Instead, we imagine K existing only in principle, and the reduction answers each query about entries of K made by the algorithm L without actually constructing K. We assume that L can make only the following types of database queries:

"Retrieve $F(s, ?)$**",** where F is one of the binary relations (functions) of type $\text{AND}_{i,j,k}$, $\text{OR}_{i,j,k}$, or $\text{NOT}_{i,k}$ as described above. This returns the value s' such that $(s, s') \in F$, that is, the individual s' that denotes the slightly-more-evaluated circuit obtained by applying function F to s. If there is no such s' (for example, if $F = \text{AND}_{i,j,k}$, but either $s[i]$ or $s[j]$ is "?", or $s[k] \neq$ "?", or k is not in the required range), then the value NIL is returned.

"Retrieve $F(?, s')$**",** where F is one of the binary relations (functions) of type $\text{AND}_{i,j,k}$, $\text{OR}_{i,j,k}$, or $\text{NOT}_{i,k}$ as described above. This returns the individual s such that $(s, s') \in F$, that is, the individual s that denotes the slightly-less-evaluated circuit which, when F is applied, yields the slightly-more-evaluated circuit s'. (If there is no such s, then the value NIL is returned. Note that for a particular choice of F and s', there is at most one s such that $(s, s') \in F$, so the query has an unambiguous answer.)

"$F(s)$**",** where F is one of the unary predicates EVAL-TO-1 or INPUT-VECTOR. This query returns TRUE if s satisfies the predicate, otherwise it returns NIL.

"Member s**",** where $s \in I$ is the name of an individual in the database. This is a standard membership query, which returns true if and only if s is a positive example of the target CLASSIC sentence. Here we note again that in this model we do not allow the construction of a hypothetical individual to be used in a membership query.

Note that the answers to each of the queries (except Member) can be computed efficiently from the description of K given above, and without requiring an explicit representation of all of the relations in K. Further note that the answers to these queries (except Member) are independent of the particular circuit B, hence can be answered without knowledge of B.

Allowing significantly more complicated queries can result in questions that are not answerable efficiently, so even if the database were available in its entirety, L could not easily obtain the answers. For example, consider a query that involves finding the projection based on a condition of a composition of function values, as in the request to find all s for which EVAL-TO-1($(f_1(f_2(\ldots(f_{m-n}(s))))))$ holds. Since the functions f_i available correspond to all possible AND, OR, and NOT functions, the search for those s's which satisfy the condition is exactly the question of whether some corresponding circuit is satisfiable—an NP-hard problem.

Now, to learn the circuit B, L' begins running the learning algorithm L. If L requests a labeled random example, L' requests a labeled random example x of B, and from it creates the corresponding individual $s \in I$ such that $s = xy$ where y is a string of "?" symbols of length $m - n$. L' gives this example to L labeled as x was labeled by B.

If L makes a query of the form "Retrieve $F(s, ?)$" (respectively, "Retrieve $F(?, s')$") then L' simply determines whether or not s (respectively, s') is of the right form with respect to the indices specified by F, and if so, returns to L the unique slightly-more-evaluated s' (respectively, slightly-less-evaluated s) as dictated by F and the bits of s (respectively, s'). As noted above, this can be computed without knowledge of B, as the function F explicitly states the simple relationship that must exist between s and s'. If s (respectively, s') is not of the correct form, then L' returns NIL as an answer to the query of L.

Similarly, if L makes a query of the form "$F(s)$" where F is one of the unary predicates EVAL-TO-1 or INPUT-VECTOR, then L' simply replies TRUE or NIL, depending on whether or not s satisfies the predicate specified. This can be determined by simple inspection of s.

Finally, suppose that L makes a membership query on the individual $s = xy$, where x has length n and y has length $m - n$. Then, if $y \neq ?^n$ (i.e., if y does not consist entirely of "?"'s), then L' responds that s is a negative example. Otherwise, L' poses a membership query of x to the Boolean circuit membership oracle. If x is a positive example of B, then L' responds to L that $s = xy$ is a positive example, otherwise L' responds that s is a negative example.

Because all information that L' provides to L is consistent with information that L would have obtained had it run with actual database K and target CLASSIC sentence S that corresponds to B, it must in time polynomial in relevant parameters, and with probability at least $1 - \delta$, output some polynomial-time program H whose error on random unlabeled examples from I is at most ϵ,

Now, to predict the value $B(x)$ of a randomly chosen Boolean vector x, L' forms the corresponding element of I with name $s = x?^{m-n}$, and evaluates $H(s)$ to predict whether or not s is a positive example of the classic sentence S. It immediately follows that with probability at least $1 - \delta$, the error of L' on random unlabeled examples of B is at most ϵ.

Note that L' asks membership queries if and only if L does, hence learning (predicting) CLASSIC sentences without membership queries from random individuals is as hard as the prediction problem for Boolean circuits without membership queries, and if membership queries are additionally allowed, the problem remains as hard as the corresponding problem for Boolean circuits with membership queries allowed. Both of these problems are intractable given the existence of 1-way functions (Angluin & Kharitonov, 1995).

7. Unions of CLASSIC Concepts

CLASSIC lacks an OR construct. As such, let us comment briefly on a target consisting of a union of CLASSIC concepts. Here we consider a positive example any concept that is subsumed by the union of the concepts in the target. Care must be taken, however, in defining what subsumption means in the presence of a union of concepts. Two possibilities present themselves.

The first possible subsumption definition stems from the definition of subsumption for a single CLASSIC concept, which considers the set of individuals selected under an interpretation. We formalize this definition and leave the question of learnability under this definition open.

The second possible definition of subsumption, which is interesting in its own right, possesses a property called *strong compactness* (Page, 1993) in other logical systems. This strong compactness property leads immediately to two positive learning results.

7.1. First Definition

Recall that for single CLASSIC concepts c_1 and c_2, c_1 subsumes c_2 if the set of individuals denoted by c_2 is a subset of the individuals denoted by c_1 regardless of the interpretation. Reasoning analogously, the first (and perhaps more compelling) definition of subsumption for a union T of CLASSIC concepts states that a concept c is subsumed by T if for every interpretation, the set of individuals denoted by c is a subset of the union of the sets of individuals denoted by the members of T. Note that this definition of subsumption permits interesting interaction between the concepts in T. For example, under any interpretation, the union

(AND `brown` (AT-LEAST 3 `spots`)) $\cup$ (AND `brown` (AT-MOST 10 `spots`))

denotes the set of all brown individuals, regardless of the number of spots. Thus the union subsumes the concept `brown` even though neither member of the union subsumes `brown` on its own. Hence, a positive example of the union of two concepts is not necessarily a positive example of either one of them — a phenomenon that does not occur in propositional settings. Reasoning about the possibility of such interaction makes the problem of learning in this situation challenging. We leave open the question of learnability under this definition of union.

7.2. Second Definition

Our second definition of subsumption, which is of interest in its own right, is that a set T of CLASSIC concepts subsumes exactly the union of the sets of positive examples of the concepts in T. Thus, an example is positive (i.e., follows from T) if and only if it is a positive example of (i.e., follows from) one of the concepts in T. This is the strong compactness property. To distinguish this definition of union from that given above, we will refer to this as the *weak union*.

This kind of non-interaction is not new. In order to make reasoning in certain first order logic systems tractable, Dalal and Etherington (1992) consider various restrictions to the interaction permitted among the elements of a set of logical sentences. In the propositional setting, Blum and Chalasani (1992) consider *switching concepts*. Here there is actually a set of targets, and the "current" target switches from time to time from one element of the set to another. An example is classified as positive or negative according to the "current" target. Thus no interaction occurs among the elements in the set of targets. Frazier et al. (1994) define a *consistently ignorant teacher*, that models, among other things, a concept defined by the agreement of a set of target concepts: This is a teacher that has a set of targets and classifies examples as positive (negative) if all elements in the set of targets classify the example as positive (negative); in the event that some targets classify the example as positive and others classify it as negative, the teacher is ignorant about the correct classification and says "I don't know."

We sketch an argument that the weak union of CLASSIC concepts is learnable. The idea of learning a (weak) union of CLASSIC sentences using this method was suggested by Rob Schapire (private communication), and is reminiscent of other approaches (Page, 1993; Angluin et al., 1992). Let T be the set of concepts constituting the target. We maintain a set U of concepts as our hypothesis by replacing any one $u \in U$ with **Prune**$(c \times u)$ where $c \times u$ is a positive example and c is a (necessarily positive) counterexample to U. If no $u \in U$ satisfies this property, then **Prune**(c) is added to the set U. Initially, U contains only the universally positive example.

To see that this works, inductively suppose our current hypothesis U is a set $\{u_1, \ldots, u_k\}$ of concepts which are each positive examples of T such that no t in T subsumes both u_i and $u_{j \neq i}$. Let c be a (necessarily positive) counterexample to U. There are two possibilities to consider – either **Prune**(c) is added to U or some u_i is replaced by $\mathbf{Prune}(c \times u_i)$.

For any concept t, t subsumes $c \times u_i$ if and only if t subsumes c and t subsumes u_i. Thus, if **Prune**(c) is added to U then no $c \times u_i$ is a positive example, hence for no u_i is there a $t \in T$ such that t subsumes u_i and t subsumes c (and hence **Prune**(c)). The inductive hypothesis is preserved.

On the other hand, suppose there is some $u_i \in U$ such that $c \times u_i$ is a positive example, so that **Prune**$(c \times u_i)$ replaces u_i in U. For this to occur, it must be that t subsumes u_i and t subsumes c for some $t \in T$. Consider the resulting hypothesis. Inductively, there is no $t \in T$ and $u_{j \neq i} \in U$ such that t subsumes both u_i and u_j. But then, since **Prune**$(c \times u_i)$ subsumes u_i, there exists no $t \in T$ such that t subsumes both u_j and **Prune**$(c \times u_i)$. The inductive hypothesis is again preserved.

Whether **Prune**(c) is added to U or **Prune**$(c \times u_i)$ replaces u_i in U, progress is made toward some member of T without undermining the progress made by other members of U toward other members of T. The moral of the story is that a definition of union that possesses the strong compactness property admits a learning algorithm that identifies each item in the target simply by updating any item in its current hypothesis when it can, and when no update to an existing item can be made it must be the case that the new counterexample is a suitable representative for some member of the target for which we have no representative. In either case, we essentially learn in parallel the different members of T, and the total time

taken is at most the sum of the running times of $\mathbf{Learn_D}$ on each member of T. We conclude that the weak union of CLASSIC concepts under this definition is efficiently learnable:

THEOREM 6 *The weak union of* CLASSIC *concepts is learnable in the PAC model with membership queries in time polynomial in* $|T|$, $\frac{1}{\epsilon}$ $\frac{1}{\delta}$, *and* n, *the upper bound on example size, where* $|T|$ *denotes the sum of the lengths of the* CLASSIC *descriptions comprising the union.*

Given this result, let us reconsider the consistently ignorant teacher mentioned above. Such a teacher will classify a description as positive if each element of its set of targets classifies it as positive, will classify a description as negative if each element of its set of targets classifies it as negative, and will say "I don't know" if there is no agreement of classification among the elements of the target set. We wish to apply Theorem 7 of (Frazier et al., 1994), which states that if the set of concepts defined by unions of concepts from a class C is learnable, and the set of concepts defined by intersections of concepts from C is learnable, then the set of agreements of concepts in C is learnable and C is learnable from a consistently ignorant teacher. Taking the union of CLASSIC sentences to be the weak union, we have just demonstrated that the union is learnable. On the other hand, the intersection of CLASSIC sentences is simply the AND of sentences and so is itself a CLASSIC sentence whose size is simply the sum of sizes of the individual CLASSIC sentences – so the intersection is also learnable. We thus obtain the following corollary.

COROLLARY 1 CLASSIC *is learnable from a consistently ignorant teacher.*

8. Membership Query Response Errors

We have shown that we need membership queries, but how much do we depend on them? What if the classification of examples is unreliable? Such questions arise from the desire to model inaccurate advice from a teacher or expert.

In particular, we consider a setting in which an adversary is permitted, with a given probability, to foul the classification of an example. It is assumed that the classification we witness for a particular example persists so that we cannot exploit the adversary's probability constraint by asking repeatedly about that example and thereby statistically determine its correct classification. This notion is known as *persistent malicious misclassification noise*. A number of authors have investigated this and related models (Angluin & Laird, 1988; Sloan, 1988; Shackelford & Volper, 1988; Auer, 1993; Decatur, 1993; Kearns & Li, 1993; Ron & Rubinfeld, 1993; Angluin & Slonim, 1994; Angluin, 1994; Angluin & Kriķis, 1994; Sloan & Turán, 1994; Frazier et al., 1994)).

The graphs we have been manipulating admit the random construction of a number of related, but distinct, graphs having the property that either all of them are positive examples or all of them are negative examples. Exploiting this property, this section gives a general test for verifying the answer to a membership query in the presence of even a significant amount of persistent malicious misclassification noise, showing that this type of difficulty can be robustly tolerated in learning CLASSIC.

For technical reasons, we consider only *reduced* labeled equivalence graphs – labeled equivalence graphs in which every vertex having "degenerate" labels (which express the null constraint) and having out degree zero also has in degree at least two – this forces the vertex to express some non-trivial restriction in the CLASSIC description. Prohibiting vertices with degenerate labels, which have indegree 1 and outdegree 0 does not change the ability of labeled equivalence graphs to represent CLASSIC descriptions concisely, because any CLASSIC sentence whose equivalence graph contains such a vertex is also representable more concisely by the equivalence graph with the vertex removed. However, permitting such vertices does illuminate a slight difference between the semantics of labeled equivalence graphs and the semantics of CLASSIC. A given labeled equivalence graph may be a negative example of the target due to the fact that some target supported string is unsupported even if the vertex reached by this supported string imposes no other semantic constraint in terms of the equivalence of strings or the vertex label – in CLASSIC such a graph would have arisen from some subexpression stating, "All individuals in the relation r to individuals in this set are individuals in the universe." Clearly such a statement is true for every role r and every individual in the universe; eliminating such a subexpression does not change the semantics of a CLASSIC description, but the two labeled equivalence graphs representing these descriptions would be semantically different.

We now introduce our model of *persistent malicious membership query responses*. Let $r(\cdot)$ be any polynomial, and let G be any equivalence graph. The first time a membership query is made on a particular G, the teacher (adversary) flips a coin that with probability $\frac{1}{2} - \frac{1}{r(|G_*|)}$ lands `heads`. If the coin lands heads, the adversary is permitted to answer the query incorrectly if he chooses; however, if the coin lands `tails` the adversary must correctly answer the query. Thereafter, the answer to a membership query on G will be the same answer as was first given, preventing the learning algorithm from obtaining information by asking the same question more than once. Our defense against such noise resides in the following lemma:

LEMMA 3 *Let G_* be the target (equivalence graph). Then there is an algorithm* **determine-label** *that on input of any equivalence graph G, any edge e of G, and any $\delta > 0$, using a membership oracle with persistent malicious misclassification noise rate $\frac{1}{2} - \frac{1}{r(|G_*|)}$, halts in time $O(\frac{r(|G_*|)^2}{2} \ln \frac{1}{\delta})$, and with probability at least $1 - \delta$ determines the correct label of the graph $G \backslash e$ with respect to G_*.*

Proof: The algorithm **determine-label** uses the following simple idea: It is possible to create many variants of $G \backslash e$ that all have the same true classification as $G \backslash e$. By taking a majority vote of the classifications, we can with high probability determine the true classification of $G \backslash e$.[6]

Let n_{G_*} be the number of vertices in G_*. Consider any string $s = s_1 s_2 \ldots s_{k-1}$ over Σ of length $k \geq n_{G_*} + 1$. Now construct a simple path consisting of k new vertices, $v_1, \ldots, v_k$, with directed edge labeled s_i from v_i to v_{i+1} for $i < k$, Leave $v_1, \ldots, v_{k-1}$ unlabeled, but make v_k the root of the graph of an arbitrarily chosen (but not inconsistent) CLASSIC concept. Now redirect the terminus of e in G to vertex v_1.

If e was deletable, then this new graph is a positive example. If e was not deletable, then either it must be used to capture some SAME-AS constraint expressed in the target or it must be used in a path that reaches some vertex label constraint expressed in the target.[7] In either case such a constraint must occur along a path of length at most n_{G_*} within the target, as that is the number of vertices in G_*. However, the first n_{G_*} vertices of this new path express no constraint of any kind, so that redirecting e must cause some constraint of G_* to be violated. Thus, if e is not deletable, this new graph is a negative example.

Let r abbreviate $r(|G_*|)$. Form a test set T by constructing $m = \frac{r^2}{2} \ln \frac{1}{\delta}$ such graphs, each with a new path based on a distinct string s as above. (Even if $|\Sigma| = 1$, the size of the largest such graph needed will be at most $O(n_{G_*} + m)$.) By the observations above, every graph in T is a positive example if e is deletable (i.e., if $G\backslash e$ is a positive example) and every graph in T is a negative example if e is not deletable (i.e., if $G\backslash e$ is a negative example). The algorithm **determine-label** asks a (noisy) membership query for every element of the test set T. Each independently has probability at least $\frac{1}{2} + \frac{1}{r}$ of being answered correctly. By Hoeffding's inequality (Hoeffding, 1963), the probability that fewer than half of the queries for graphs in T are answered correctly is at most e^{-2m/r^2}, which by choice of m is at most δ. Thus, if the majority response is output, the probability that **determine-label** misclassifies $G\backslash e$ is at most δ. ■

We now have our most general result:

THEOREM 7 *Algorithm* **Learn$_{\mathbf{p}}$**, *augmented with algorithm* **determine-label**, *can be used to PAC-learn* CLASSIC *sentences, using random examples, and using membership queries with malicious persistent classification noise rate* $1/2 - 1/r(|G_*|)$. *The algorithm runs in time polynomial in* $|G_*|$, $r(|G_*|)^2$, $\frac{1}{\delta}$, $\frac{1}{\epsilon}$, *and the length* n *of counterexamples, and outputs a* CLASSIC *sentence that has error with respect to* G_* *at most* ϵ, *with probability at least* $1 - \delta$.

Proof: Run **Learn$_{\mathbf{p}}$** with parameters ϵ and $\delta/2$, noting that membership queries are used only in the **Prune** procedure of Figure 5, and that each such query concerns a graph G with edge e to be removed. Instead of asking a single membership query on $G\backslash e$ to the noisy membership oracle, run algorithm **determine-label** with inputs G, e, and $\frac{\delta}{2s}$, where s is the total number of membership queries that **Learn$_{\mathbf{p}}$** would make with noise-free membership queries. The probability that *any* of the invocations of this procedure is incorrect totals at most $\delta/2$. The probability that the hypothesis produced by **Learn$_{\mathbf{p}}$** has error exceeding ϵ is at most δ, which includes the event of probability at most $\delta/2$ that some response of **determine-label** is inaccurate. ■

Note that if an algorithm **determine-label** could be constructed for arbitrary queries (and not just those resulting from deleting an edge from some G), then we could employ a variant of **Learn$_{\mathbf{p}}$** in the PAC setting with all examples maliciously mislabeled: we could simply ignore the labels of examples, and instead apply **determine-label** to obtain the correct classification. There are subtle technical reasons why this cannot be done in any obvious way, although a variety of techniques similar to the one used by **determine-label**

are available. We leave open the question of whether an algorithm exists for determining the label of arbitrary examples.

9. Summary

We have demonstrated a positive polynomial time learnability result using membership and equivalence queries for labeled equivalence graphs with vertex labels chosen from a finite lattice, and we adapted this algorithm to obtain a polynomial time algorithm for the natural first-order concept class CLASSIC. We then showed that the learnability did not rest solely on the power of the membership queries by giving a non-learnability result for membership query only algorithms. An alternative model of learning from individuals was investigated, and shown to be as hard as learning arbitrary Boolean circuits, hence intractable assuming the existence of one-way functions.

We concluded by examining extensions to the positive result by considering two different kinds of unreliability – a non-omniscient teacher and a malicious adversary. Learnability in the former setting followed easily from a particular possible definition of the meaning of a union of CLASSIC concepts. In the latter, robust learnability in the presence of random unreliable responses to membership queries was presented in the PAC learning model. Surprisingly, it was shown that even highly unreliable membership queries are not dispensable.

Our work ended leaving the following questions open. Is there an algorithm to correctly determine the labeling of arbitrary concepts using membership queries answered with a high misclassification noise rate? Our results relied on the ability to determine the correct labeling only of concepts whose equivalence graphs are missing at least one edge. Extending the result to arbitrary equivalence graphs would allow a result that CLASSIC is learnable in the PAC setting from *unlabeled* examples, provided that a (highly noisy) membership query oracle is available. Is the union of CLASSIC concepts learnable under the more compelling definition of the meaning of the union?

More generally, what other types of knowledge representations are efficiently learnable, and what types of queries are necessary? Are fairly general, and practical, knowledge representations learnable using natural queries to an expert? Can we help open the "knowledge acquisition" bottleneck in expert system design?

Acknowledgments

We thank William Cohen and Haym Hirsh for invaluable discussions and clarifications regarding CLASSIC. Any correct statement about CLASSIC found in this paper is surely the consequence of their patience. Any incorrect statements are of our own design. William also pointed out the application of the consistently ignorant teacher model to the problem of CLASSIC learning. The referees made a number of helpful comments which improved the readability of the paper, and Tom Hancock, Tom Dietterich, and Karen Cullen at Kluwer demonstrated patience that went far beyond the call of duty.

Notes

1. Keep in mind that these two different "types" of individuals are indistinguishable within a description logic statement; it is the venue of the externally supplied meanings of the roles and primitives to preserve any intuitive distinctions we may have concerning these different "types" of individuals.
2. Along with e remove any unreachable component.
3. They show how to exactly learn, with equivalence queries only, the cross-product of t intervals over $[1..m]$ in time polynomial in t and $\log m$.
4. To adhere to the semantics of CLASSIC, consider a collection of roles all of which are attributes.
5. The membership problem for a concept class C is the following: Given (the description of) a $c \in C$, and an example x, determine whether or not x is a positive or negative example of c.
6. The construction assumes the language contains a role symbol. If there are no role symbols, the graphs of the concepts expressible consist of but a single vertex. Such a concept class is learnable without any membership queries, by finding the upper bound of the vertex labeling of all positive examples in a sufficiently large (but polynomially sized) random sample. This single vertex algorithm can be interleaved with the construction about to be presented.
7. This makes use of the assumption that the equivalence graph is reduced.

References

Angluin, D. (1987). Learning regular sets from queries and counterexamples. *Information and Computation, 75(2),* 87–106.

Angluin, D. (1988a). Learning with hints. *Proceedings of the 1988 Workshop on Computational Learning Theory* (pp. 167–181). San Mateo, CA: Morgan Kaufmann.

Angluin, D. (1988b). Queries and concept learning. *Machine Learning, 2(4),* 319–342.

Angluin, D. (1988c). *Requests for hints that return no hints*. Technical report YALEU/DCS/RR-647, Yale University.

Angluin, D. (1992). Computational learning theory: survey and selected bibliography. *Proceedings of the Twenty Fourth Annual ACM Symposium on Theory of Computing* (pp. 351–369). Victoria, BC: ACM Press.

Angluin, D. (1994). *Exact learning of μ-DNF formulas with malicious membership queries*. Technical Report YALEU/DCS/TR-1020, Yale University.

Angluin, D., Frazier, M., & Pitt, L. (1992). Learning conjunctions of Horn clauses. *Machine Learning, 9,* 147–164.

Angluin, D. & Kharitonov, M. (1995). When won't membership queries help? *Journal of Computer and System Sciences, 50*.

Angluin, D. & Kriķis, M. (1994). Learning with malicious membership queries and exceptions. *Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory* (pp. 57–66). New York: ACM Press.

Angluin, D. & Laird, P. (1988). Learning from noisy examples. *Machine Learning, 2(4),* 343–370.

Angluin, D. & Slonim, D. K. (1994). Randomly fallible teachers: learning monotone DNF with an incomplete membership oracle. *Machine Learning, 14(1),* 7–26.

Auer, P. (1993). On-line learning of rectangles in noisy environments. *Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory* (pp. 253–261). New York: ACM Press.

Beck, H., Gala, H., & Navathe, S. (1989). Classification as a query processing technique in the CANDIDE semantic model. *Data Engineering Conference* (pp. 572–581). Los Angeles, CA.

Blum, A. & Chalasani, P. (1992). Learning switching concepts. *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory* (pp. 231–242). New York: ACM Press.

Blumer, A., Ehrenfeucht, A., Haussler, D., & Warmuth, M. K. (1989). Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM, 36(4),* 929–965.

Bobrow, D. G. & Winograd, T. (1977). An overview of KRL, a knowledge representation language. *Cognitive Science, 1(1),* 3–46.

Borgida, A. (1992). Description logics are not just for the `flightless-birds`: a new look at the utility and foundations of description logics. Preprint.

Borgida, A., Brachman, R. J., McGuinness, D. L., & Resnick, L. (1989). CLASSIC: A structural data model for objects. *Proceedings of SIGMOD-89* Portland, Oregon.

Borgida, A. & Patel-Schneider, P. F. (1992). *A semantics and complete algorithm for subsumption in the CLASSIC description logic*. Technical report, AT&T.

Brachman, R. J., Fikes, R. E., & Levesque, H. J. (1983). Krypton: A functional approach to knowledge representation. *IEEE Computer, 16(10),* 67–73.

Chen, Z. & Maass, W. (1994). On-line learning of rectangles and unions of rectangles. *Machine Learning, 17(2/3),* 23–50.

Cohen, W. (1993a). Cryptographic limitations on learning one-clause logic programs. *Proceedings of the Tenth National Conference on Artificial Intelligence.* Washington, D.C.

Cohen, W. (1993b). Pac-learning a restricted class of recursive logic programs. *Proceedings of the Tenth National Conference on Artificial Intelligence.* Washington, D.C.

Cohen, W. W., Borgida, A., & Hirsh, H. (1992). Computing least common subsumers in description logics. *Proceedings of the Tenth National Conference on Artificial Intelligence*.

Cohen, W. W. & Hirsh, H. (1994a). Learnability of description logics with equality constraints. *Machine Learning, 17(2/3),* 169–199. Special issue on Computational Learning Theory: COLT'92.

Cohen, W. W. & Hirsh, H. (1994b). Learning the CLASSIC description logic: Theoretical and experimental results. *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference (KR94)*: Morgan Kaufmann.

Cohen, W. W. & Page, C. D. Jr. (1994). Polynomial learnability and inductive logic programming: Methods and results. Submitted for publication.

Dalal, M. & Etherington, D. (1992). Tractable approximate deduction using limited vocabulary. In *CSCSI-92* Vancouver.

Decatur, S. E. (1993). Statistical queries and faulty PAC oracles. *Proceedings of the Sixth Annual ACM Conference Computational Learning Theory* (pp. 262–268). New York: ACM Press.

Devanbu, P., Brachman, R. J., Selfridge, P., & Ballard, B. (1991). LaSSIE: A knowledge-based software information system. *Communications of the ACM, 35(5).*

Dietterich, T. G., London, B., Clarkson, K., & Dromey, G. (1982). Learning and inductive inference. In *The Handbook of Artificial Intelligence, Volume 3*. William Kaufmann.

Džeroski, S., Muggleton, S., & Russell, S. (1992). Pac-learnability of logic programs. *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory* (pp. 128–135). New York: ACM Press.

Frazier, M., Goldman, S., Mishra, N., & Pitt, L. (1994). Learning from a consistently ignorant teacher. *Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory* (pp. 328–339). New York: ACM Press. To appear, *Information and Computation*.

Frazier, M. & Page, C. D. (1993). Learnability of recursive, non-determinate theories: Some basic results and techniques. *Third International Workshop on Inductive Logic Programming.*

Frazier, M. & Pitt, L. (1993). Learning from entailment: An application to propositional horn sentences. *Proceedings of the Tenth International Conference on Machine Learning* (pp. 120–127). San Mateo, CA: Morgan Kaufmann.

Haussler, D. (1989). Learning conjunctive concepts in structural domains. *Machine Learning, 4(1),* 7–40.

Haussler, D., Littlestone, N., & Warmuth, M. K. (1994). Predicting $\{0, 1\}$ functions on randomly drawn points. *Information and Computation, 115(2),* 284–293.

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association, 58(301),* 13–30.

Kearns, M. & Li, M. (1993). Learning in the presence of malicious errors. *SIAM Journal on Computing, 22,* 807–837.

Kearns, M. & Valiant, L. G. (1994). Cryptographic limitations on learning Boolean formulae and finite automata. *Journal of the ACM, 41(1),* 67–95.

Littlestone, N. (1988). Learning when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning, 2,* 285–318.

Mays, E., Apte, C., Griesmer, J., & Kastner, J. (Fall 1987). Organizing knowledge in a complex financial domain. *IEEE Expert,* (pp. 61–70).

Muggleton, S. (1991). Inductive logic programming. *New Generation Computing, 8,* 295–318.

Page, C. D. & Frisch, A. M. (1992). Generalization and learnability: A study of constrained atoms. In S. H. Muggleton (Ed.), *Inductive Logic Programming* chapter 2. London: Academic Press.

Page, C. D. (1993). *Anti-unification in constraint logics: Foundations and applications to learnability in first-order logic, to speed-up learning, and to deduction*. PhD thesis, University of Illinois at Urbana-Champaign.

Patel-Schneider, P. F. (1989). A four-valued semantics for terminological logics. *Artificial Intelligence, 38,* 319–351.

Pitt, L. & Warmuth, M. K. (1990). Prediction preserving reducibility. *Journal of Computer and System Sciences, 41(3),* 430–467. Special issue for the *Third Annual Conference of Structure in Complexity Theory*.

Rivest, R. L. & Schapire, R. E. (1993). Inference of finite automata using homing sequences. *Information and Computation, 103(2),* 299–347.

Ron, D. & Rubinfeld, R. (1993). Learning fallible finite state automata. *Proceedings of the Sixth Annual ACM Conference Computational Learning Theory* (pp. 218–227). New York: ACM Press.

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning, 5(2),* 197–227.

Shackelford, G. & Volper, D. (1988). Learning k-DNF with noise in the attributes. *Proceedings of the 1988 Workshop on Computational Learning Theory* (pp. 97–103). San Mateo, CA: Morgan Kaufmann.

Sloan, R. (1988). Types of noise in data for concept learning. *Proceedings of the 1988 Workshop on Computational Learning Theory* (pp. 91–96). San Mateo, CA: Morgan Kaufmann.

Sloan, R. H. & Turán, G. (1994). Learning with queries but incomplete information. *Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory* (pp. 237–245). New York: ACM Press.

Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM, 27(11),* 1134–1142.

Received March 9, 1995
Accepted July 27, 1995
Final Manuscript July 15, 1996