Machine Learning, 17, 169–199 (1994)

The Learnability of Description Logics with Equality Constraints

WILLIAM W. COHEN wcohen@research.att.com
AT&T Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974

HAYM HIRSH hirsh@cs.rutgers.edu
Department of Computer Science, Rutgers University, New Brunswick, NJ 08903

Editor: Lisa Hellerstein

Abstract. Although there is an increasing amount of experimental research on learning concepts expressed in first-order logic, there are still relatively few formal results on the polynomial learnability of first-order representations from examples. Most previous analyses in the pac-model have focused on subsets of Prolog, and only a few highly restricted subsets have been shown to be learnable. In this paper, we will study instead the learnability of the restricted first-order logics known as "description logics", also sometimes called "terminological logics" or "KL-ONE-type languages". Description logics are also subsets of predicate calculus, but are expressed using a different syntax, allowing a different set of syntactic restrictions to be explored. We first define a simple description logic, summarize some results on its expressive power, and then analyze its learnability. It is shown that the full logic cannot be tractably learned. However, syntactic restrictions exist that enable tractable learning from positive examples alone, independent of the size of the vocabulary used to describe examples. The learnable sublanguage appears to be incomparable in expressive power to any subset of first-order logic previously known to be learnable.

Keywords: Pac-learning, concept learning, first-order representations, description logics

1. Introduction

1.1. Motivation and prior work

Recently, there has been an increasing amount of experimental research on learning concepts expressed in first-order logic (Muggleton & Feng, 1992; Quinlan, 1990; Muggleton, 1992). Unfortunately, there are relatively few formal results on the polynomial learnability of first-order representations from examples; although a number of first-order learning systems have been formally analyzed (Shapiro, 1982; Banerji, 1988; Muggleton & Buntine, 1988) most of these analyses either allow powerful queries, such as membership and subset queries, or allow exponentially slow convergence rates, as in Gold's (1967) model of "learnability in the limit". To date only a handful of formal results have been obtained for first-order representations in Valiant's (1984) model of pac-learnability.

Prior results will be discussed in detail in Section 6; briefly, however, the strongest positive result known to date is due to Džeroski, Muggleton, and Russell (1992). They show that a single nonrecursive function-free constant-depth "determinate" definite clause is pac-learnable against any distribution; they also show that a non-recursive logic program containing a constant number of such clauses is learnable against certain distributions.

Some very recent work has shown that if any of these restrictions are relaxed, then the resulting language is not pac-learnable (Kietz, 1993) and is not even learnable in the weaker model of *polynomial predictability* (Cohen, 1993a). An earlier negative result is due to Haussler (1989), who analyzed a first-order language containing existential quantification and conjunction; this language is closely related to function-free definite clauses. Haussler showed that even under severe syntactic restrictions this language is not pac-learnable from examples alone.

In summary, previous analyses in the pac-model have focused primarily on subsets of Prolog, and have discovered only a few highly restricted first-order languages that are learnable. In this paper, we will take a different approach, and study instead the first-order languages called *description logics (DLs)*.

Description logics (sometimes also called *terminological logics*) are a family of formalisms for representing knowledge. DLs have found applications in several areas, ranging from database interfaces (Beck et al., 1989), to software information bases (Devanbu et al., 1991) to financial management (Mays et al., 1987). Historically, they are descended from the KL-ONE language (Brachman, 1979), which in turn is closely related to *semantic nets* (Quillian, 1967); KL-ONE in fact began as an effort to provide a precise logical foundation for semantic nets. Description logics are also related to *frame-based* representation systems (Bobrow & Winograd, 1977; Minsky, 1975), but differ in placing a greater emphasis on declarative representation. Recent surveys of research on DLs can be found in (Borgida, 1992; MacGregor, 1991; Woods & Schmolze, 1992).

Over the years, a number of experimental learning systems have used semantic net or frame-based representations. Winston's (1975) thesis work is one well-known example; other more recent researchers have also used DLs as the underlying representation (Morik, 1989; Vilain et al., 1990; Beck, 1991; Conklin & Gasglow, 1992). To date, however, there has been little formal analysis of the learnability of such representations. In undertaking such an analysis, description logics are a natural starting point for several reasons. First, DLs are a declarative formalism, closely tied to logic, and the formal aspects of reasoning with description logics have received considerable attention from the research community. As a consequence, the semantics of DLs and the complexity of deductive reasoning with DLs are quite well-understood. A second and related advantage is that many implemented DLs have been carefully designed so that certain types of deductive reasoning, notably taxonomic reasoning, can be done efficiently; this raises the possibility that non-deductive reasoning (*e.g.*, learning) can also be done efficiently.

A final advantage of DLs is that although DLs correspond to restricted subsets of first-order predicate calculus, DLs have a radically different syntax. The differences in syntax suggest alternative ways of syntactically restricting expressive power—alternatives, that is, to the syntactic restrictions considered by Haussler (1989) and Džeroski, Muggleton, and Russell (1992)—and hence may lead to new and different learnable subsets of first-order logic. Indeed, one of the results of this paper is a description of a learnable DL that appears to be incomparable in expressive power to any subset of first-order logic previously known to be learnable.

1.2. Contributions of the paper

Much of the research in computational learning theory seeks to characterize the class of learnable languages. The primary technique for this task is to identify representational *boundaries of learnability:* an example of such a boundary provided by the result given by Kearns and Valiant (1989) showing that unrestricted propositional logic is not learnable in the Valiant model, together with various positive results on the learnability of restricted subsets of propositional logic (Kearns et al., 1987; Rivest, 1987). The high-level goal of this paper is to apply this methodology to DLs, with the aim of discovering how much representational power can be incorporated in a DL while still allowing pac-learnability.

In Section 2 we briefly define the formal models of learnability used in this paper. Section 3 then introduces a DL called CORECLASSIC. This DL was chosen because, although it is relatively simple, it nonetheless illustrates several of the more subtle issues involved in pac-learning more complex DLs. It is furthermore a subset of CLASSIC, a DL that has been implemented and used in several real-world applications (Beck et al., 1989; Devanbu et al., 1991; Mays et al., 1987; Schewe, 1989). Section 3 also contains several examples of CORECLASSIC descriptions, and a brief discussion of how CORECLASSIC differs from first-order representations based on definite clauses.

Section 4 then shows that CORECLASSIC cannot be pac-learned. This result also gives some insight as to why pac-learning is difficult. Based on these insights, we propose a restriction on CORECLASSIC, and show that given this additional restriction, CORECLASSIC is learnable in a very strong sense: in particular, the restricted language is learnable from positive examples alone, using a number of examples independent of the size of the vocabulary used to describe examples. Our positive result can thus be viewed as an extension to a first-order language of the result of Blum (1990) on learning monomials over infinite attribute spaces. Finally, we will discuss related formal results on learning first-order concepts in somewhat more detail, and conclude.

2. Models of learnability

2.1. Preliminary definitions

Our first model of learnability is a slight variant of *pac-learnability*, as introduced by Valiant (1984). Let X be a set, called the *domain*. Define a *concept* C over X to be any subset of X, and a *language* $\mathcal{L}$ to be a set of concepts. Associated with $\mathcal{L}$ is some scheme for representing the concepts in $\mathcal{L}$. In general, we will be quite casual about the distinction between a concept and its representation; when there is a risk of confusion we will write the set denoted by a representation C (*i.e.*, the extension of C) as $ext(C)$. We will assume a *size measure* on representations $C \in \mathcal{L}$, which will be written $\|C\|$, and we will use $\mathcal{L}_n$ to denote $\{C \in \mathcal{L} : \|C\| \leq n\}$. If P is a probability distribution, a *sample of C drawn according to P* is a pair of multisets S^+, S^- drawn according to P, S^+ containing only the positive examples of C, and S^- containing the negative ones.

So far, this is precisely as in Valiant's model. However, we differ in our definition of "example." First, in our learning problems, the size of examples may vary: the

parameter n_e is used to denote the size of the largest example, and we define a sample to be n_e-*bounded* if it contains no example larger than n_e.[1] Second, while in Valiant's model an example is a member of the domain X, in our model an example is a concept in $\mathcal{L}_{n_e}$. We will precisely state the meaning of examples later in Definition 3.3: roughly, for a target concept C, an example $x \in \mathcal{L}_{n_e}$ is counted as positive if $ext(x) \subseteq ext(C)$ and negative if $ext(x) \not\subseteq ext(C)$.

This "single representation trick" (Dietterich et al., 1982)—*i.e.*, assuming that examples are described in the same language as concepts—is common in AI machine learning systems, and has also been used several times previously in the formal literature (Pitt & Warmth, 1990; Haussler, 1989). In the context of first-order learning, it is useful because there is no standard representation for instances: thus assuming examples are themselves concepts eliminates the need to introduce a separate language for examples.[2]

2.2. *Pac-learnability*

The model of pac-learnability was introduced by Valiant (1984) and has since been extensively studied. Informally, pac-learnability requires that the learning algorithm *LEARN* outputs an accurate hypothesis from a given language $\mathcal{L}$ most of the time, whenever the target concept is succinctly expressible in $\mathcal{L}$.

More formally, we define a language $\mathcal{L}$ to be *pac-learnable* iff there is an algorithm *LEARN* and a polynomial function $m(\frac{1}{\epsilon}, \frac{1}{\delta}, n_e, n_t)$ so that for every $n_t > 0$, every $C \in \mathcal{L}_{n_t}$, every $0 < \epsilon < 1$, every $0 < \delta < 1$, and every probability distribution P, *LEARN* has the following behavior: when run on a n_e-bounded sample S^+, S^- of C drawn according to P of size $|S^+| + |S^-| > m(\frac{1}{\epsilon}, \frac{1}{\delta}, n_e, n_t)$, *LEARN* outputs a hypothesis $H \in \mathcal{L}$ such that $Prob(P(H \triangle C) > \epsilon) < \delta$ where $\triangle$ denotes symmetric difference. The probability above is taken over the possible samples S^+ and S^- and (if *LEARN* is a randomized algorithm) over any coin flips made by *LEARN*; furthermore, *LEARN* runs in time polynomial in $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, n_e, n_t, and the size of the sample.

2.3. *Mistake-bounded identifiability*

Our second model of learnability is much stronger. If C is a concept in $\mathcal{L}$, define a n_e-*bounded positive presentation of* C to be a sequence $x_1, \ldots, x_i, \ldots$ of positive examples of C such that for all i, $\|x_i\| \leq n_e$. An *incremental learning algorithm for* $\mathcal{L}$ is an algorithm *ILEARN* which reads in a positive presentation one element at a time, and after reading in each x_i, outputs a hypothesis $H_i \in \mathcal{L}$. Finally, we say that $\mathcal{L}$ is *mb-identifiable with one-sided error* if there is an incremental learning algorithm *ILEARN* for $\mathcal{L}$ and a polynomial function $b(n_t, n_e)$ such that for every $C \in \mathcal{L}$, and for every positive presentation of C, *ILEARN* has the following properties:

- No hypothesis H_i output by *ILEARN* ever errs on negative data (*i.e.*, $ext(H_i) \subseteq ext(C)$).
- There are most $b(n_t, n_e)$ examples x_i that are misclassified by H_{i-1}.

- *ILEARN* takes time polynomial in n_t and n_e to process each example x_i.

The function $b(n_t, n_e)$ is called the *mistake bound* of the learner. Informally, the mistake bound is simply the worst-case number of mistakes that the learning algorithm can make in learning a target concept of size n_t from examples of size n_e.

This model combines elements of Valiant's model of pac-learning with one-sided error (Valiant, 1984) and Littlestone's model of mistake-bounded learnability (Littlestone, 1988), and is stronger than either: from the Valiant model it adopts the constraints that no errors occur on negative examples and that the hypothesis must be represented as a concept in $\mathcal{L}$, while from the Littlestone model, it adopts the constraint that learning is incremental and makes a bounded number of mistakes. This strong model of learnability will be used only in positive results.

3. The CORECLASSIC description logic

Our results are concerned with the DL CORECLASSIC. CORECLASSIC is a subset of CLASSIC, a DL that has been implemented and used in several real-world applications (Beck et al., 1989; Devanbu et al., 1991; Mays et al., 1987; Schewe, 1989). We chose this subset because it accurately illustrates the complexities that arise in learning DLs: in particular, both the learning algorithm of Section 5 and the hardness results of Section 4 extend to full CLASSIC (Cohen & Hirsh, 1992). Because this representation language is both nontrivial and unfamiliar to many readers, this background section is rather lengthy.

3.1. *Descriptions in* CORECLASSIC

In CORECLASSIC *descriptions* describe subsets of a domain X, the elements of which are called *individuals*. This section gives a formal definition of CORECLASSIC descriptions; we follow this definition with some extended examples.

Descriptions are built from an alphabet containing symbols of two types. *Primitive class symbols* $\mathtt{p}_1, \ldots, \mathtt{p}_{n_p}$ denote specific subsets of X; the $\mathtt{p}_i$ can thus be viewed as unary predicates ranging over X.[3] For example, `person`, `company`, or `college-graduate` might be primitive class symbols. *Role symbols* $\mathtt{r}_1, \ldots, \mathtt{r}_{n_r}$ denote specific subsets of $X \times X$; the $\mathtt{r}_i$ can thus be viewed as binary predicates over $X \times X$. If $\mathtt{r}$ denotes a function,[4] then it is called an *attribute*; in this paper attributes will usually be denoted by the letters `a` or `b`. For example, `employee` might be a role (typically relating a member of the primitive class `company` to a member of the primitive class `person`), and `mother` might be an attribute (typically relating two `persons`).

A *description* over these two alphabets is defined below. A more detailed discussion of the semantics of descriptions is provided by Borgida and Patel-Schneider (1994).

Definition. A CORECLASSIC *description over the primitive alphabet* $\mathcal{P}rim = \{\mathtt{p}_1, \ldots, \mathtt{p}_{n_p}\}$ *and role alphabet* $\mathcal{R}ole = \{\mathtt{r}_1, \ldots, \mathtt{r}_{n_r}\}$ is defined recursively as follows.

- Any primitive $\mathtt{p}_i$ is a description; it represents the set of all elements of X that are in the set $\mathtt{p}_i$. For example, if `person` is a primitive, then `person` is also a description.
- If D is a description and $\mathtt{r}_i$ is a role, then (**ALL** $\mathtt{r_i}$ D) is a description; it represents the set of all elements $x \in X$ for which $\mathtt{r}_i(x, y)$ is true only if y is an element of description D. For example, (**ALL** `employee person`) is a description.
- If $D_1, \ldots, D_k$ are descriptions then (**AND** $\mathtt{D_1}$ ... $\mathtt{D_k}$) is a description; it represents the intersection of the k descriptions. For example,

  (**AND** `company` (**ALL** `employee person`))

  is a description.
- If $\mathtt{a}_1, \ldots, \mathtt{a}_k$ and $\mathtt{b}_1, \ldots, \mathtt{b}_l$ are attributes, then

  (**SAME-AS** $(\mathtt{a}_1 \ldots \mathtt{a}_k)$ $(\mathtt{b}_1 \ldots \mathtt{b}_l)$)

  is a description; it represents the set of all elements $x \in X$ for which

  $$\mathtt{a}_k(\cdots \mathtt{a}_2(\mathtt{a}_1(x)) \cdots) = \mathtt{b}_l(\cdots \mathtt{b}_2(\mathtt{b}_1(x)) \cdots)$$

  For example, (**SAME-AS** `(address) (spouse address)`) is a description.

The following additional examples of descriptions should help make the CORECLASSIC language more tangible.

Example: To describe the set of women whose daughters are all unemployed theory PhDs, and all of whose sons are married to doctors, one might use the description

```
(AND woman
     (ALL daughter (AND theoryPhD unemployed))
     (ALL son (AND married (ALL spouse doctor))))
```

□

Example: To describe the set of mortgage applications that are guaranteed by the applicant's mother-in-law, one might use the description

```
(AND mrtgapplic
     (SAME-AS (guarantor) (applicant spouse mother)))
```

□

We also add to the syntax defining CORECLASSIC the following shorthands: (**AND**) denotes X, the entire domain, and (**ALL** $(\mathtt{r}_1\ \mathtt{r}_2 \ldots \mathtt{r}_n)$ D) abbreviates (**ALL** $\mathtt{r}_1$ (**ALL** $\mathtt{r}_2$ ... (**ALL** $\mathtt{r}_n$ D) ...)). As a special case of this, (**ALL** () D) abbreviates D. Finally, the *size* $\|D\|$ of a description D is defined to be the number of primitive and role symbols appearing in the description.

3.2. CORECLASSIC *and logic programs*

As noted in the introduction, many experimental systems that learn first-order concepts use some subset of Prolog as a representation language. We will now briefly and informally compare CORECLASSIC descriptions with those concepts that can be expressed with first-order definite clauses.

A typical representation for concepts is that used by FOIL (Quinlan, 1990). FOIL learns concepts represented by a set of Prolog clauses containing no function symbols; for instance FOIL might learn the following clause:

```
p(X) ← r(X,Y),q(Y).
```

This clause represents the set of all objects X for which the clause above succeeds if given to a Prolog interpreter:[5] that is, it represents the set

$$\{X : \exists Y\ (r(X,Y) \wedge q(Y))\} \tag{1}$$

This concept cannot be expressed in CORECLASSIC, as CORECLASSIC has no way of representing the existential quantification on Y. There are also other constructs available in Prolog that are not available in CORECLASSIC (for example, direct use of predicates of arity greater than two); this example, however, is sufficient to demonstrate that CORECLASSIC is not strictly more expressive than function-free definite clauses.

On the other hand, CORECLASSIC can express concepts such as (**ALL** r q), which corresponds to the set

$$\{X : \forall Y\ (r(X,Y) \rightarrow q(Y))\} \tag{2}$$

This is clearly different from the set of Equation 1. Furthermore, there does not appear to be any direct way to convert this to a function-free Prolog program over the predicates r and q; for example, if one uses the technique suggested above to encode the set of Equation 2 and constructs the predicate

$$\mathtt{p(X)} \leftarrow \forall \mathtt{Y}[\mathtt{r(X,Y)} \rightarrow \mathtt{q(Y)}]$$

and then converts to clausal form in the usual way, the result is neither function-free nor definite. This suggests that CORECLASSIC and function-free Prolog are incomparable.

A more rigorous comparison of CORECLASSIC and function-free Prolog appears elsewhere (Borgida, 1993). In Section 6 we will consider the question of how learnable DLs compare in expressive power with the learnable subsets of Horn clause logic.

3.3. *Description graphs and subsumption*

We will now review some relevant formal results on CORECLASSIC. First, many of our proofs will use an equivalent representation for CORECLASSIC descriptions called *concept graphs*, introduced by Borgida and Patel-Schneider (1994) as a tool for formalizing another subset of CLASSIC called "basic CLASSIC".[6] A CORECLASSIC concept

graph is a rooted labeled directed graph $(V, E, v_0, \ell_V(*))$, where V are the vertices, $E \subseteq (V \times V \times \mathcal{R}ole)$ are the edges, and $\ell_V(v)$ is a function that associates a set of primitive class symbols with a vertex v. If $(v, w, \mathtt{r}) \in E$ we will say that $\mathtt{r}$ is a *role label* (or, when clear from context, simply a *label*) of an edge from v to w. A *path* through a concept graph G is a sequence of edges $((v_0, v_1, r_0), (v_1, v_2, r_1), \ldots, (v_{k-1}, v_k, r_{k-1}))$. The *size* $\|G\|$ of a concept graph G is the sum of the number of edges and the cardinalities of the vertex labels. The semantics of concept graphs are given by the following definition.

Definition. Given a graph $G = (V, E, v_0, \ell_V(*))$, and a vertex $v \in V$, an individual x is in the *extension of* v iff the following are all satisfied:

- for each $\mathtt{p}_i \in \ell_V(v)$, $\mathtt{p}_i(x)$ is true;
- for each edge $(v, w, \mathtt{r}_i) \in E$, if y is any individual such that $\mathtt{r}_i(x, y)$, then y is in the extension of w.
- for each pair of non-looping paths starting at v, labeled only with attributes, and ending at the same vertex of G, following these two chains of attributes from x leads to the same individual x'.

Finally, x is in the *extension of* G iff it is in the extension of the root vertex v_0.

Although it will require a few more constraints on the nature of concept graphs, we will shortly show that for every CORECLASSIC description there is an equivalent concept graph, and vice versa. The basic idea will be that vertex labels in a concept graph will correspond to conjunctions of primitives, edges will correspond to **ALL**s, and two paths leading to the same vertex will correspond to **SAME-AS** conditions.[7]

The following properties of concept graphs will be important in showing the equivalence of CORECLASSIC descriptions and concept graphs. A concept graph is *deterministic* if no vertex has two outgoing edges with the same role label. This is not an important restriction, as there is a polynomial-time algorithm for making a concept graph deterministic; this algorithm is described in the proof of Theorem 1 below. A concept graph is *well-formed (with respect to equalities)* if the following property holds: whenever there are two distinct non-looping paths p_1 and p_2 from vertex v to vertex w, there are three strings α, β and γ such that

- the sequence of role labels on the edges of p_1 corresponds to the string $\gamma\alpha$,
- the sequence of role labels on the edges of p_2 corresponds to the string $\gamma\beta$, and
- α and β contain only attribute symbols.

The intuition behind well-formedness is that, for any two paths from v labeled only with attributes and converging to the same vertex, if the graph is well-formed then the equality implicitly expressed by the two paths can be expressed by a CORECLASSIC description, namely the description

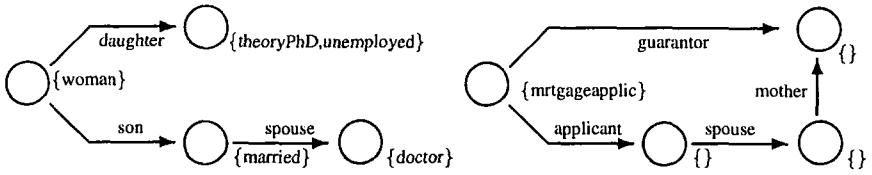


Figure 1. Two examples of concept graphs

$$(\textbf{ALL}\ (r_1 \ldots r_m)\ (\textbf{SAME-AS}\ (a_1 \ldots a_k)\ (b_1 \ldots b_l)))$$

where $\gamma = r_1 \ldots r_m$, $\alpha = a_1 \ldots a_k$, and $\beta = b_1 \ldots b_k$. (Recall from Section 3.1 that we use the syntax (**ALL** (r_1 r_2 ... r_n) D) to abbreviate (**ALL** r_1 (**ALL** r_2 ... (**ALL** r_n D) ...)).)

Example: To make the equivalence of CORECLASSIC descriptions and concept graphs tangible, consider the two CORECLASSIC descriptions used as examples in Section 3. Figure 1 contains concept graphs equivalent to the two descriptions. (The leftmost vertex of each graph is the root; this convention will be followed throughout this paper.) Both of these graphs are deterministic and well-formed. □

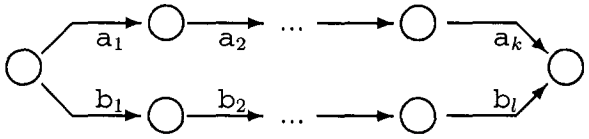
The existence of concept graphs equivalent to the descriptions of Example 3.1 is no accident, as the following theorem shows.

THEOREM 1 *For every* CORECLASSIC *description D, there is a semantically equivalent deterministic well-formed concept graph of size polynomial in $\|D\|$ that can be constructed in polynomial time; for every well-formed concept graph G, there is a semantically equivalent* CORECLASSIC *description of size polynomial in $\|G\|$ that can be constructed in polynomial time.*

Proof: The first part of the theorem is Theorem 1 of Borgida and Patel-Schneider (1994), which (in order to keep this paper self contained) we will now summarize. The proof is based on a recursive algorithm that converts a CORECLASSIC description to an equivalent concept graph:

- If the description is a primitive p_i, then construct a concept graph which consists of a single vertex labeled with the set $\{p_i\}$.
- If the description is of the form (**ALL** r_i D'), then first construct a concept graph G' equivalent to D'. Then, add to G' a single vertex v labeled with the empty set, and a single edge labeled r_i from v to the root of G'; finally make the new vertex v the root of the modified graph. This graph is equivalent to the description (**ALL** r_i D').

- If the description is of the form (**SAME-AS** $(\mathtt{a}_1 \ldots \mathtt{a}_k)$ $(\mathtt{b}_1 \ldots \mathtt{b}_l)$), then construct a concept graph of this form:



- If the description is of the form (**AND** D_1 D_2), again recursively construct concept graphs G_1 and G_2 equivalent to D_1 and D_2 respectively. A graph G that is equivalent to (**AND** D_1 D_2) can be formed by taking the union of the vertex and edge sets of G_1 and G_2 and then *merging* the roots of G_1 and G_2: to *merge* two vertices v_1 and v_2 in a graph, one forms of a new vertex v_{12} such that the label of v_{12} is the union of the labels of v_1 and v_2, and such that every edge leaving (or entering) either v_1 or v_2 has been redirected to leave (or enter) v_{12}. The merger of the two roots becomes the root of the constructed graph.

The result of this process is a graph that has the right extension and is well-formed with respect to equivalence; however the graph is not necessarily deterministic. To make the graph deterministic, the following operation is used: First, find some vertex v that has two outgoing edges (v, w_1, r) and (v, w_2, r), then merge the vertices w_1 and w_2. This operation is repeated until the graph is deterministic. We omit the arguments for the correctness of this procedure.

To prove the second part of the theorem, we will describe a procedure that generates a CORECLASSIC description equivalent to a given deterministic well-formed concept graph. First, construct a spanning tree for the concept graph. The desired description is the conjunction of the following descriptors:

- For every vertex v, there will be one conjunct of the form (**ALL** $(\mathtt{r}_{v,1} \ldots \mathtt{r}_{v,k_v})$ (**AND** $\mathtt{p}_{v,1} \ldots \mathtt{p}_{v,l_v}$)) where $\mathtt{r}_{v,1}, \ldots, \mathtt{r}_{v,k_v}$ are the labels on the path in the spanning tree to vertex v, and $\{\mathtt{p}_{v,1}, \ldots, \mathtt{p}_{v,l_v}\}$ is the label of v.
- For each edge $(v, w, \mathtt{r})$ not in the spanning tree, there will be one conjunct of the form

  (**ALL** $(\mathtt{r}_1 \ldots \mathtt{r}_m)$ (**SAME-AS** $(\mathtt{a}_1 \ldots \mathtt{a}_k)$ $(\mathtt{b}_1 \ldots \mathtt{b}_l)$))

  where $\mathtt{r}_1 \ldots \mathtt{r}_m$, $\mathtt{a}_1 \ldots \mathtt{a}_k$, and $\mathtt{b}_1 \ldots \mathtt{b}_k$ are as follows. Let p_1 be the path in the spanning tree from the root to w, and let p_2 be a path from the root to w via v and the edge $(v, w, \mathtt{r})$; notice that these are two distinct non-looping paths from a single node (the root) to w. The description above is derived from the strings $\gamma = \mathtt{r}_1 \ldots \mathtt{r}_m$, $\alpha = \mathtt{a}_1 \ldots \mathtt{a}_k$, and $\beta = \mathtt{b}_1 \ldots \mathtt{b}_k$ guaranteed by the definition of well-formedness. (Note that γ, α and β can be found in polynomial time by simply letting γ be the largest common prefix of p_1 and p_2.)

To prove the correctness of this procedure, it is sufficient to show that applying Borgida and Patel-Schneider's procedure for converting a description to concept graph will produce a graph isomorphic to the original one. We sketch a proof of this as follows. First,

notice that the first set of conjuncts, when conjoined, produce the spanning tree, along with the correct labels for its vertices (this fact follows by induction on the algorithm given above for constructing a graph from a description.) Also, each conjunct from the last set, when it is conjoined with the spanning tree and the resulting graph is made deterministic, will have the effect of adding the corresponding non-tree edge. Thus the resulting graph is isomorphic to the original graph. ■

Example: As an example of the procedure for converting a concept graph to a description, consider the first graph of Figure 3.3. As this graph is a tree, the spanning tree must be the graph itself, yielding the set of conjuncts

```
(ALL () (AND woman))
(ALL daughter (AND theoryPhD unemployed))
(ALL son (AND married))
(ALL (son spouse) (AND doctor))
```

with no additional **SAME-AS** conjuncts since there are no edges absent from the spanning tree. The conjunction of these yields a CORECLASSIC description equivalent to that of the example.

For the second graph of Figure 1, imagine that the spanning tree chosen contains all of the edges of the graph except for the one labeled "mother". The first set of conjuncts generated by the procedure would be

```
(ALL () (AND mrtgageapplic))
(ALL guarantor (AND))
(ALL applicant (AND))
(ALL (applicant spouse) (AND))
```

which will generate the spanning tree when conjoined, and the second set of conjuncts would be

```
(ALL () (SAME-AS (applicant spouse mother) guarantor))
```

Again, the conjunction of all five conjuncts yields a description equivalent to the second example of Figure 1. □

Given this result, we will focus on learnability using the more analytically tractable concept graph representation. However, we first briefly consider the tractability of an important operation for description logics: testing *subsumption.* D_1 is said to *subsume* D_2 if membership in D_2 implies membership in D_1 regardless of the way in which the primitive classes and roles are defined. More formally:

Definition. Let D_1 and D_2 be two descriptions (or concept graphs). D_1 is said to *subsume* D_2 (written $D_2 \Longrightarrow D_1$) iff $ext(D_1) \supseteq ext(D_2)$ for all possible definitions of the primitives and roles used in D_1 and D_2.

This allows us to state precisely how examples are treated.

Definition. For a target concept C, an example $x \in \mathcal{L}_{n_c}$ is *positive* if $x \Longrightarrow C$ and *negative* if $x \not\Longrightarrow C$.

Although CORECLASSIC has limited expressiveness, its restrictions are *not* imposed for learnability reasons: rather, they are imposed so that subsumption checking is a tractable operation:

THEOREM 2 (BORGIDA & PATEL-SCHNEIDER) *Let D_1 and D_2 be two* CORECLASSIC *descriptions (or well-formed concept graphs). It can be decided in polynomial time if $D_2 \Longrightarrow D_1$.*

Proof: It is convenient to assume that D_2 has been converted to a concept graph G_2. To test if a description of the form (**ALL** $(\mathtt{r}_1 \ldots \mathtt{r}_k)$ (**AND** $\mathtt{p}_1 \ldots \mathtt{p}_l$)) subsumes G_2, it is sufficient to start at the root of G_2 and follow the path labeled $\mathtt{r}_1 \ldots \mathtt{r}_k$. If at any point this path cannot be followed (*i.e.*, if at some point there is no edge corresponding to the next label) then the description does not subsume G_2. Otherwise, let the vertex v be the destination of the path; G_2 is subsumed iff $\ell_V(v) \supseteq \{\mathtt{p}_1 \ldots \mathtt{p}_l\}$.

To test if a description of the form

(**ALL** $(\mathtt{r}_1 \ldots \mathtt{r}_\mathtt{m})$ (**SAME-AS** $(\mathtt{a}_1 \ldots \mathtt{a}_\mathtt{k})$ $(\mathtt{b}_1 \ldots \mathtt{b}_\mathtt{l})$))

subsumes G_2, it is sufficient to perform the following test. First, start at the root of G_2 and follow the path $\mathtt{r}_1 \ldots \mathtt{r}_m$ to its destination vertex v. Then follow the two paths $\mathtt{a}_1 \ldots \mathtt{a}_k$ and $\mathtt{b}_1 \ldots \mathtt{b}_l$ from v, and test to see if they end at the same vertex of G_2. If so, then G_2 is subsumed; if the two paths end at different vertices, or if any of the three paths cannot be followed through G_2, then G_2 is not subsumed.

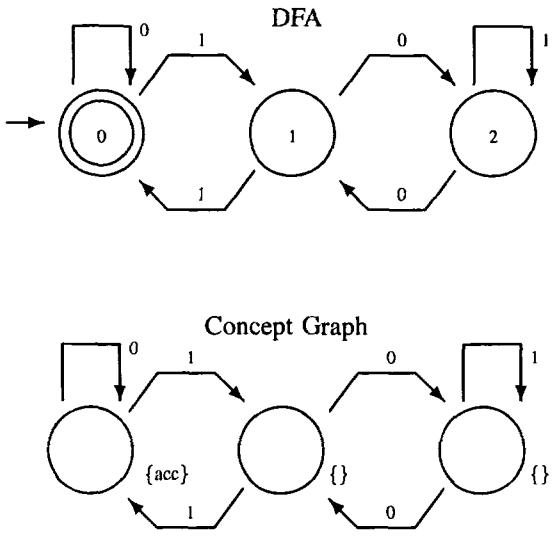
Finally, notice that by converting a description to a concept graph and back again using the technique of Theorem 1, any description can be put in a normal form that is a conjunction of descriptions of the two types described above; to test if a description of the form (**AND** $D_1 \ldots D_k$) subsumes G_2, it is sufficient to check that each D_i subsumes G_2. ■

4. A negative result

This section presents the first result of this paper, namely that CORECLASSIC, even under two severe syntactic constraints, cannot be pac-learned unless RP=NP.[8] Both this result and the positive result of the next section utilize a correspondence between deterministic finite automata (DFAs) and CORECLASSIC concept graphs. A presentation of this key idea thus begins this section.

4.1. CORECLASSIC *descriptions and DFAs*

We will begin with a demonstration of the fact that CORECLASSIC descriptions can be used to emulate DFAs. This should not be surprising: note that at least superficially,



Description

(**AND** (**ALL** 0 acc)
(**SAME-AS** (0) ()) (**SAME-AS** (1 1) ()))
(**SAME-AS** (1 0 0) (1)) (**SAME-AS** (1 0 1) (1 0))

Figure 2. A DFA and an equivalent CORECLASSIC description

CORECLASSIC concept graphs are quite similar to DFAs, as both can be represented as rooted deterministic directed graphs with labeled nodes and edges.

To make this statement more precise, consider the following proposition.[9]

PROPOSITION 1 *For every DFA* $M = (Q, \Sigma, \delta, q_0, F)$ *there is a deterministic concept graph* G *over the primitive alphabet* $\{\mathtt{acc}\}$ *and role alphabet* Σ *(where every* $a \in \Sigma$ *is an attribute) such that*

$$a_1 \ldots a_n \in L(M) \text{ iff } G \Longrightarrow (\mathbf{ALL}\ (\mathtt{a}_1\ \ldots\ \mathtt{a}_n)\ \mathtt{acc})$$

In other words, for every DFA M there is a concept graph G such that checking to see if a string x is accepted by M is equivalent to a particular subsumption test involving G.

To verify the proposition, let us simplify the way transition graphs of a DFA are represented by deleting rejecting states (and any incident edges) that have no accepting

successor states. Now let G be a concept graph isomorphic to the simplified transition graph of the DFA, except that accepting states q_{acc} have the label $\ell_V(q_{acc}) = \{\texttt{acc}\}$ and rejecting states q_{rej} have the empty label $\ell_V(q_{rej}) = \emptyset$. (Note that since we have made every symbol from Σ an attribute, every graph will be well-formed.) An example of this construction is shown in Figure 2. The correctness of this proposition is immediate from this construction and the subsumption algorithm of Theorem 2.

The nature of this connection between DFAs and CORECLASSIC descriptions is perhaps made a little clearer by the following observation. Consider two concept graphs G_1 and G_2 with corresponding DFAs M_1 and M_2 such that $G_1 \Longrightarrow G_2$. Since subsumption is transitive, for any $a_1 \ldots a_n \in L(M_2)$

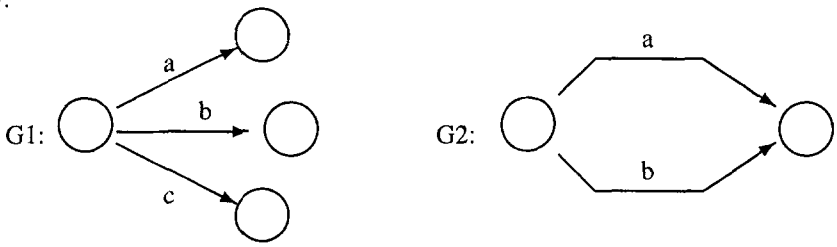
$$
\begin{array}{lll}
 & a_1 \ldots a_n \in L(M_2) & \\
\text{implies} & G_2 \Longrightarrow (\textbf{ALL}\ (\texttt{a}_1\ \ldots\ \texttt{a}_n)\ \texttt{acc}) & \text{by Proposition 1} \\
\text{implies} & G_1 \Longrightarrow (\textbf{ALL}\ (\texttt{a}_1\ \ldots\ \texttt{a}_n)\ \texttt{acc}) & \text{since } G_1 \Longrightarrow G_2 \\
\text{implies} & a_1 \ldots a_n \in L(M_1) & \text{by Proposition 1}
\end{array}
$$

In other words, any string in $L(M_2)$ must also be in $L(M_1)$. We therefore have the following fact:

$$\text{if } G_1 \Longrightarrow G_2 \text{ then } L(M_2) \subseteq L(M_1) \tag{3}$$

Thus, there is a close connection between concept graph subsumption and regular language inclusion, although it is the reverse of what one might expect: if G_2 subsumes (roughly, is a superset of) G_1, then the regular language corresponding to G_2 is a *subset* of the language corresponding to G_1. This is easiest to remember if one thinks of a concept graph as representing a set of paths, each of which is a restriction on concept membership.

It should be noted that the converse of Equation 3 does *not* hold. In particular, it may be that $L(M_2) \subseteq L(M_1)$ and $G_1 \not\Longrightarrow G_2$; this would be true if there were **SAME-AS** conditions implied by G_2 that do not hold in G_1, as is the case in the graphs shown below:



Because of this, and because of the fact that the correspondence between CORECLASSIC and DFAs leads to an inverted containment relationship, Proposition 1 does not have any immediate corollaries regarding the learnability of CORECLASSIC. The proposition is still useful, however, for two reasons. First, we will be making heavy use of CORECLASSIC descriptions with DFA-like properties in the proof of our negative result. Second, understanding the relationship between CORECLASSIC and DFAs may help the reader in developing intuitions about the expressive power of the language.

4.2. A pac-learning result

We will now show that CORECLASSIC is not pac-learnable, even under two severe syntactic constraints.

THEOREM 3 *Assuming that RP≠NP,* CORECLASSIC *is not pac-learnable (regardless of the size of the hypothesis the learner is allowed to produce) even if either of the following constraints hold:*

- *the primitive class alphabet is singleton, the role alphabet is doubleton, and the concept graph of every example is acyclic, or*
- *the primitive class alphabet is singleton, and the concept graph of every example contains only three vertices.*

Proof: There are two parts to the proof, one for each set of constraining circumstances, and each part follows the standard argument used by Pitt and Valiant (1988). In particular, in each section of the proof, we reduce the NP-hard problem 3SAT (Hopcroft & Ullman, 1979) to a consistency problem: given a 3-CNF formula ϕ we construct a polynomial-sized set of positive and negative examples such that ϕ is satisfiable iff a hypothesis consistent with the examples exists. By making an adversarial choice of a distribution and letting the error rate ϵ be small enough, we can create a learning problem which forces any pac-learner to output a hypothesis consistent with the data with high probability, thus solving the 3SAT problem in randomized polynomial time.

To generate the appropriate positive and negative examples we will first construct DFAs for the given SAT problem, then use the construction of the previous subsection to convert these DFAs into concept graphs that will serve as the necessary training data.

Let $\phi = \bigwedge_{i=1}^{m}(l_{i_1} \vee l_{i_2} \vee l_{i_3})$ be a 3CNF formula over at most $3m$ variables. Without loss of generality, we assume that for all i the literals l_{i_1}, l_{i_2}, and l_{i_3} are in strictly increasing alphabetical order (*i.e.*, $i_1 < i_2 < i_3$), and that all variables appear in ϕ.

The first construction. We will construct a series of positive examples $G_{P_1}, \ldots, G_{P_m}$ (from associated DFAs $M_{P_1}, \ldots, M_{P_m}$), each of which corresponds to a clause of ϕ, and a single negative example G_N (from an associated DFA M_N). Intuitively, each G_{P_i} will ensure that any consistent hypothesis G_H (when interpreted as a DFA—it is easy to verify that any such hypothesis must be so interpretable) will accept only strings that (when interpreted as assignments to the variables of ϕ) satisfy the i-th clause. G_N will simply ensure that the DFA corresponding to G_H accepts some string—*i.e.*, that its language is not empty. Together these constraints will force the hypothesis to represent some satisfying assignment for ϕ.

More precisely, $G_{P_1}, \ldots, G_{P_m}$ and G_N will be defined so that ϕ is satisfiable iff a concept graph G_H exists with the following properties:

Property 1. $\forall i : 1 \leq i \leq m$, $G_{P_i} \Longrightarrow G_H$.

(*I.e.*, G_H is consistent with the positive examples.)

Property 2. $G_N \not\Longrightarrow G_H$.

(*I.e.*, G_H is consistent with the negative examples.)

Let $s(l)$ denote the "sign" of the literal l; *i.e.*, $s(l)$ is 1 if $l = x_i$ and $s(l)$ is 0 if $l = \overline{x_i}$. Without loss of generality, assume that the two attribute symbols are 0 and 1. We can thus define M_N to accept the empty set $\emptyset$, and M_{P_i} to be the minimal DFA accepting the language

$$\begin{array}{l}(0+1)^{i_1-1}s(l_{i_1})(0+1)^{3m-i_1} \ + \\ (0+1)^{i_2-1}s(l_{i_2})(0+1)^{3m-i_2} \ + \\ (0+1)^{i_3-1}s(l_{i_3})(0+1)^{3m-i_3}\end{array}$$

The binary strings of length $3m$ accepted by these DFAs can be interpreted as assignments to the variables that appear in ϕ. Notice that M_{P_i} accepts exactly the assignments that satisfy the i-th clause of ϕ; also, if any description G_H subsumes G_{P_i} then it cannot include a **SAME-AS** condition involving strings of length i_1, i_2, or i_3.

Now, assume ϕ is satisfiable, let w be a binary string encoding a satisfying assignment, and let M_H be the minimal DFA that accepts the language $\{w\}$. Clearly G_H satisfies Property 2. Since $L(M_H) \subseteq L(M_{P_i})$ for each i and G_H contains no **SAME-AS** conditions, G_H also satisfies Property 1.

Conversely, assume that a G_H (with corresponding M_H) satisfying Properties 1 and 2 above exists. Since $G_N \not\Longrightarrow G_H$ it must be that either $L(M_H) \not\subseteq L(M_N)$, or that G_H contains a **SAME-AS** condition not true in G_N. The latter case is impossible: since every variable x_i appears in some clause C_j, and since G_H subsumes each G_{P_j}, G_H cannot include **SAME-AS** conditions involving strings of length i for any i. In the former case, it must be that $L(M_H) \neq \emptyset$. Let w be some word in $L(M_H)$. From Property 1 above, for each i, $G_{P_i} \Longrightarrow G_H$, hence $L(M_H) \subseteq L(M_{P_i})$, and thus $w \in L(M_{P_i})$. It therefore follows that w encodes a satisfying assignment of ϕ. Thus ϕ is satisfiable iff an G_H satisfying Properties 1 and 2 exists.

To complete the proof that learning is hard in the first set of circumstances, consider a distribution that is uniform over all of the examples. Note that if a pac-learner exists, then for any $\epsilon < \frac{1}{m+1}$ the pac-learner must in time polynomial in m almost always output a hypothesis consistent with all of the examples. The pac-learner can thus be used to decide in polynomial time if a satisfying assignment exists. This completes the first construction.

The second construction. In the second construction, rather than representing an assignment to the variables of ϕ as a bit vector, we will represent an assignment as a "complete monomial": *i.e.*, a monomial that contains either x_j or $\overline{x_j}$ for each variable x_j, and never contains both x_j and $\overline{x_j}$. The alphabet for our DFAs is thus the set $\Sigma = \{x_1, \overline{x_1}, \ldots, x_{3m}, \overline{x_{3m}}\}$. We will again use the negative example G_N where M_N accepts the empty set, but we will introduce two sorts of "positive" DFAs. First, for each variable x_j, we define M_{P_j} to be a DFA accepting the language

$$(x_1 + \overline{x_1} + \ldots + x_{j-1} + \overline{x_{j-1}})^*(x_j + \overline{x_j})(x_{j+1} + \overline{x_{j+1}} + \ldots + x_{3m} + \overline{x_{3m}})^*$$

but *not* the minimal such DFA; instead we insist that strings containing x_j and strings containing $\overline{x_j}$ reach distinct accepting states. Notice that M_{P_j} forces any hypothesis that contains x_j or $\overline{x_j}$ to contain only one of the two literals, and also forces that literal to appear just once. Furthermore, any hypothesis subsuming M_{P_j} cannot contain any equalities involving the symbols x_j or $\overline{x_j}$. Thus it is easy to show that any hypothesis G_H consistent with all the G_{P_i} must be a tree in which each path is a list of literals, one per variable, in increasing order, terminating in a node labeled with `acc`; thus each path in G_H is a complete monomial.

Let us extend the notation for regular expressions slightly: when $S = \{w_1, \ldots, w_n\}$ is a set of words, we will let S^* denote the set $(w_1 + \ldots + w_n)^*$. To construct the second set of positive examples, for each clause C_i in ϕ, define M_{P_i} to be the minimal DFA accepting

$$(\Sigma - C_i)^*(l_{i_1} + l_{i_2} + l_{i_3})\Sigma^*$$

M_{P_i} accepts exactly those strings that, if they are in fact assignments, satisfy the clause C_i. It can be easily verified that all of these "example DFAs" require at most three states to implement. (Of course, in contrast to the previous reduction, this result requires a number of role symbols proportional to the number of examples.)

The remainder of the argument precisely parallels that of the previous construction, except with $\epsilon < \frac{1}{4m+1}$. ■

5. A positive result

Given that CORECLASSIC is not pac-learnable, we now consider the learnability of restricted sublanguages. Two plausible restrictions are to constrain concept graphs to be acyclic, or to restrict the number of vertices in a concept graph; however, the result of Theorem 3 indicates that neither of these restrictions alone makes efficient learning possible. A natural question to ask is whether CORECLASSIC is learnable if both of these restrictions are made, *i.e.*, if all examples are acyclic concept graphs with a bounded number of vertices. The answer to this question is affirmative: this sublanguage is learnable even under the very strong model of mistake-bounded identifiability with one-sided error.

In this section, we develop a more general version of this result. In particular, we will define a restricted sublanguage that includes not only acyclic concept graphs with a bounded number of vertices, but also all concept descriptions that do not use the **SAME-AS** construct, and several other interesting subsets of CORECLASSIC. In developing this result, we will start with a specific learning algorithm, and then introduce syntactic restrictions that make the algorithm tractable.

5.1. An algorithm for learning from positive examples

In any conjunctively closed language $\mathcal{L}$, there is a natural algorithm for learning from positive examples only: given a set of positive examples $x_1, \ldots, x_m$, return the most specific

Let $G_1 = (V_1, E_1, v_{01}, \ell_{V_1}(*))$ and $G_2 = (V_2, E_2, v_{02}, \ell_{V_2}(*))$ be two deterministic concept graphs. Then

$$\mathrm{LCS}(G_1, G_2) = (V_{\mathrm{LCS}}, E_{\mathrm{LCS}}, v_{0\mathrm{LCS}}, \ell_{V_{\mathrm{LCS}}}(*))$$

where V_{LCS}, E_{LCS}, $v_{0\mathrm{LCS}}$, $\ell_{V_{\mathrm{LCS}}}(*)$ are defined as follows:

- $v_{0\mathrm{LCS}} = (v_{01}, v_{02})$
- V_{LCS} = the subset of $(V_1 \times V_2)$ that is reachable from $v_{0\mathrm{LCS}}$ with the edge set E_{LCS} defined below.
- $E_{\mathrm{LCS}} = \{((v_1, v_2), (w_1, w_2), \mathtt{r}) : (v_1, w_1, \mathtt{r}) \in E_1 \text{ and } (v_2, w_2, \mathtt{r}) \in E_2\}$.
- $\ell_{V_{\mathrm{LCS}}}((v_1, v_2)) = \ell_{V_1}(v_1) \cap \ell_{V_2}(v_2)$.

Figure 3. An algorithm for computing the least common subsumer

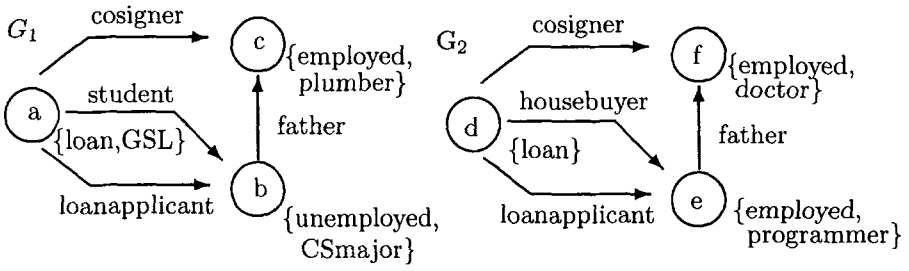
concept in $\mathcal{L}$ that includes these examples. This basic idea has surfaced in a large number of contexts; in computational learning theory, such algorithms have been variously called ordering algorithms (Natarajan, 1987) or closure algorithms (Helmbold et al., 1990), in experimental machine learning, the terms maximally specific conjunctive generalization (Dietterich & Michalski, 1983) and most specific generalization (Hirsh, 1990) have been used, and in the context of first-order clauses the terms (relative) least general generalization (Buntine, 1988; Plotkin, 1969) and anti-unification (Pfenning, 1991; Idestam-Almquist, 1993) are common.

In our learning framework, set inclusion has been replaced by the $\Longrightarrow$ relationship, so the natural analog is to return the least concept (in the lattice imposed by $\Longrightarrow$) that subsumes all of the examples. Following previous terminology used in the knowledge representation community (Cohen et al., 1992) we will call this concept the *least common subsumer (LCS)* of the examples.

Figure 3 describes an algorithm for computing the LCS of two concept graphs. The algorithm is a slight extension of the algorithm for constructing the intersection of two regular languages encoded as DFAs (Hopcroft & Ullman, 1979, page 59); the only change is that the rules for generating labels on nodes of a graph are different.

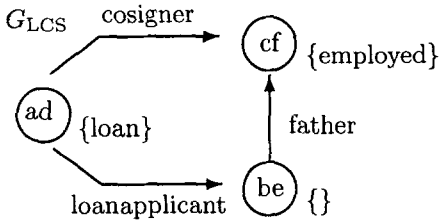
Shortly, we will present a proof of correctness for this algorithm; first, however, we will illustrate it with an example.

Example: Consider the operation of the algorithm of Figure 3 given the two concept graphs shown in Figure 4. The edges of G_{LCS} are defined to be $E_{\mathrm{LCS}} = \{((v_1, v_2), (w_1, w_2), \mathtt{r}) : (v_1, w_1, \mathtt{r}) \in E_1 \text{ and } (v_2, w_2, \mathtt{r}) \in E_2\}$; thus for each edge in the LCS graph there must be two edges in G_1 and G_2 that share a common label. This observation makes it easy to see that E_{LCS} contains only the following three edges:




```
(AND (AND loan GSL)
     (ALL cosigner
       (AND employed plumber)
     (ALL loanapplicant
       (AND unemployed CSmajor)
     (SAME-AS (loanapplicant)
              (student))
     (SAME-AS (cosigner)
              (student father)))
```

```
(AND (AND loan mortgage)
     (ALL cosigner
       (AND employed doctor)
     (ALL loanapplicant
       (AND employed programmer)
     (SAME-AS (loanapplicant)
              (housebuyer))
     (SAME-AS (cosigner)
              (housebuyer father)))
```




```
(AND loan
     (ALL cosigner employed)
     (SAME-AS (cosigner)
              (loanapplicant father))
```

Figure 4. Two concept graphs and their least common subsumer

$E_{\mathrm{LCS}} = \{$ ((a,d),(b,e),loanapplicant),
((a,d),(c,f),cosigner),
((b,e),(c,f),father) $\}$

V_{LCS} is that subset of $V_1 \times V_2$ which is reachable from the root vertex (a, d) using these edges. Thus

$V_{\mathrm{LCS}} = \{(\mathrm{a}, \mathrm{d}), (\mathrm{b}, \mathrm{e}), (\mathrm{c}, \mathrm{f})\}$

Finally, the label of a vertex (v_1, v_2) in G_{LCS} is the intersection of the label of v_1 in G_1 and the label of v_2 in G_2. Thus

$\ell_{V_{\mathrm{LCS}}}((\mathrm{a}, \mathrm{d})) = \{\mathrm{loan}\}$
$\ell_{V_{\mathrm{LCS}}}((\mathrm{b}, \mathrm{e})) = \emptyset$
$\ell_{V_{\mathrm{LCS}}}((\mathrm{a}, \mathrm{d})) = \{\mathrm{employed}\}$

The concept graph for G_{LCS} and its equivalent description are also shown in Figure 4. □

We will now prove that this algorithm does indeed compute the LCS of two concept graphs, and also give an upper bound on the size of the LCS computed by the algorithm.

THEOREM 4 *Let G_1 and G_2 be two well-formed deterministic concept graphs. Then the LCS of G_1 and G_2 exists, is unique*[10] *and well-formed, has size $\|\mathrm{LCS}(G_1, G_2)\| \leq \|G_1\| \cdot \|G_2\|$, and is computed by the algorithm of Figure 3.*

Proof: We begin by proving the last statement: that the LCS algorithm of Theorem 4 returns a least common subsumer of G_1 and G_2. To prove this, we must show first, that the concept graph G_{LCS} returned by the LCS algorithm is a common subsumer and second, that there is no other more specific common subsumer.

To show that G_{LCS} is a common subsumer, it is enough (by symmetry) to show that G_{LCS} subsumes G_1. Let us define a *rooted path* in a concept graph G to be any path that starts at the root of G; note that a rooted path p in G can be denoted by a sequence of vertices $v_{0,p}, \ldots, v_{n,p}$ where $v_{0,p}$ is the root of G. We define a rooted path p_1 in G_1 to *path-subsume* a rooted path p_2 in G_2 if all of the following hold:

- path p_2 is at least as long as p_1;
- the edges that link consecutive vertices in p_1 have the same label as the corresponding edges in p_2;
- for each vertex v_{i,p_1} in p_1, if there are two distinct paths from v_{i,p_1} labeled with role symbols $a_1 \ldots a_k$ and $b_1 \ldots b_l$ that terminate in the same vertex in G_1 then there are two distinct paths with the same labels from the corresponding vertex v_{i,p_2} in p_2; and
- each vertex v_{i,p_1} in p_1 is labeled with a subset of the label of the corresponding vertex in p_2; *i.e.*, for each i,

$$\ell_V(v_{i,p_1}) \subseteq \ell_V(v_{i,p_2})$$

Making use of the definitions of subsumption and the extension of a concept graph, it is possible to show the following statement by induction:

PROPOSITION 2 *Concept graph G_1 subsumes concept graph G_2 iff for every rooted path p_1 in G_1, there is a rooted path p_2 in G_2 such that p_1 path-subsumes p_2.*

It can now be verified easily that G_{LCS} subsumes G_1 by noting that for every rooted path p_{LCS} in G_{LCS}, there is a rooted path p_1 in G_1 such that p_{LCS} path-subsumes p_1. To see this, consider a path $p_{\mathrm{LCS}} = v_{0,\mathrm{LCS}}, \ldots, v_{n,\mathrm{LCS}}$ in G_{LCS}. Recall that each $v_{i,\mathrm{LCS}}$ was formed from pairing two vertices (v_{i,p_1}, v_{i,p_2}) where v_{i,p_1} is a vertex in G_1 and v_{i,p_2} is a vertex in G_2, and consider the path $p_1 = v_{0,p_1}, \ldots, v_{n,p_1}$ in G_1. Note that $v_{i-1,\mathrm{LCS}}$ and $v_{i,\mathrm{LCS}}$ are connected by an edge labeled $\mathtt{r}$ only if v_{i-1,p_1} and v_{i,p_1} are connected by an edge labeled $\mathtt{r}$; this immediately establishes the first three conditions for p_{LCS} to path-subsume p_1. To establish the final condition, recall that the label of each $v_{i,\mathrm{LCS}}$ is defined as

$$\ell_V(v_{i,\mathrm{LCS}}) = \ell_V(v_{i,1}) \cap \ell_V(v_{i,2})$$

and hence must be a subset of the label of $v_{i,1}$.

An argument similar to the one above shows that if G_1 is well-formed, then G_{LCS} is also well-formed.

To show that G_{LCS} is a *least* common subsumer, we must show that no more specific concept graph subsumes both G_1 and G_2. Notice that there are three ways to make G_{LCS} more specific: one could add a new rooted path, specialize an existing rooted path by adding a new primitive to some vertex label, or add a new equality to the graph (*i.e.*, a new pair of paths that begin at some vertex v and lead to the same vertex w, representing a **SAME-AS** condition.) It can be readily seen that each of these operations is impossible:

- The construction of G_{LCS} guarantees that p is a rooted path in G_{LCS} only if it is a rooted path in both G_1 and G_2; thus if any path is added to G_{LCS}, one of G_1 and G_2 will no longer be subsumed by G_{LCS}.

- Similarly, recall that each vertex $v_{i,\mathrm{LCS}}$ in G_{LCS} that was formed from pairing two vertices $v_{i,1}$ from G_1 and $v_{i,2}$ from G_2 is labeled with the intersection of the labels of these vertices; *i.e.*, $\ell_V(v_{i,\mathrm{LCS}}) = \ell_V(v_{i,1}) \cap \ell_V(v_{i,2})$. If an additional primitive is added to the label of $v_{i,\mathrm{LCS}}$, then this primitive must not be a member of the label of either $v_{i,1}$ or $v_{i,2}$. Assume without loss of generality that it is not a member of $v_{i,1}$; then the path to $v_{i,\mathrm{LCS}}$ no longer path-subsumes the path to $v_{i,1}$, and since G_{LCS} is deterministic, *no* path path-subsumes this path, and the specialization of G_{LCS} is no longer a common subsumer.

- Finally, consider merging vertices of G_{LCS} so that a new equality condition holds: *i.e.*, a new pair of paths p_1, p_2 that begin at some vertex $v_{i,\mathrm{LCS}}$ and lead to the same vertex w_{LCS}. Again, it is easy to show that if this pair of paths was not in G, then that pair of paths is not in one of G_1 or G_2.

We conclude that G_{LCS} is a least common subsumer.

Finally, we note that G_{LCS} must be the unique least common subsumer. To see this, assume that a second incomparable least common subsumer G'_{LCS} exists, and consider the conjunction of these concepts: it is easy to show that this conjunction is both a common subsumer of G_1 and G_2 and more specific than G_{LCS}, contradicting the result above. ■

5.2. *A restricted sublanguage*

Since LCS is a least upper bound operation on a lattice, it is commutative and associative, and thus the LCS of a set of m concept graphs can be found by a series of pairwise LCS operations. Hence the algorithm of Figure 3 also gives a means of computing a hypothesis consistent with a set of data. However, since the LCS of two graphs can grow as the product of their sizes, the LCS of a set of m graphs of size n_e can grow exponentially (as the results of Section 4 suggest). In looking for learnable subsets of CORECLASSIC, one approach is to consider cases for which the LCS of a set of concept graphs can be tractably computed.

To this end, let us denote the set of all paths in G as *paths*(G). (Thus *paths*(G) for a cyclic graph is infinite.) We have the following result:

LEMMA 1 *If $G_1, \ldots, G_m$ are deterministic concept graphs, then the number of vertices in* $\mathrm{LCS}(G_1, \ldots, G_m)$ *is no greater than* $min_i(|paths(G_i)|)$.

Proof: Let $G_{\mathrm{LCS}} = \mathrm{LCS}(G_1, G_2)$. Every vertex v in G_{LCS} is reachable by some path $(v_{01}, v_{02}) \ldots (v_{k1}, v_{k2})$ in G_{LCS}. Since for every edge from (v_{i1}, v_{i2}) to (v_{j1}, v_{j2}) in G_{LCS} there is an edge from v_{i1} to v_{j1} in G_1, for every vertex v there is also a path $v_{01} \ldots v_{k1}$ in G_1. Hence G_{LCS} cannot have more vertices than G_1 has paths. By symmetry, the same holds for G_2, and hence the number of vertices in G_{LCS} is bounded by

$$\min(|paths(G_1)|, |paths(G_2)|)$$

The statement of the theorem now follows easily by induction on m. ■

Finally, notice that no role symbols appear unless they are used in every G_i, and hence $\|G_i\|$ is also an upper bound on the number of role symbols in the LCS; since no vertex has more than this number of outgoing edges, we have the following corollary.

COROLLARY 1 *If $G_1, \ldots, G_m$ are deterministic concept graphs, then*

$$\|\mathrm{LCS}(G_1, \ldots, G_m)\| \leq (\min_i \|G_i\|) \cdot (\min_i |paths(G_i)|)$$

An interesting sublanguage to consider is the following.

Definition. [k-CORECLASSIC] For any constant k, k-CORECLASSIC is the set of CORECLASSIC concept graphs G such that $|paths(G)| \leq \|G\|^k$.

It follows from the results above that for any fixed constant k, the LCS algorithm can be tractably applied to a series of m positive examples, and will always output a hypothesis consistent with the (unused) negative examples.

We note also that there are at least two sufficient conditions for membership in k-CORECLASSIC which are easy to test.

- First, any acyclic concept graph must be in k-CORECLASSIC if its depth is bounded by k.
- Second, and perhaps more interestingly, any description which has an acyclic concept graph must be in k-CORECLASSIC if it contains no more than k **SAME-AS** conditions.[11]

  A special case of this, of course, are descriptions which have no **SAME-AS** conditions. These correspond to concept graphs which are trees; for concept trees the LCS algorithm above generates a new tree containing the largest common prefix of its inputs.

In most applications of CLASSIC, descriptions with cyclic concept graphs or with large numbers of **SAME-AS** conditions are rare; thus this restriction seems to be reasonable from a practical standpoint. Notice that k-CORECLASSIC descriptions may be of arbitrary size, and (if use of **SAME-AS** is appropriately restricted) may also be of arbitrary depth.

Given these results, a positive learnability result could be generated using standard techniques (*i.e.*, bound the VC-dimension of k-CORECLASSIC, and then apply the results of Blumer *et al.* (1989) to show that it is pac-learnable). Instead we will argue for the pac-learnability of this class by a slightly different technique; this will yield a somewhat stronger result, as discussed at this end of this section.

The key to the positive learnability result will be the incremental learning algorithm LCSLEARN:

Definition. Let $x_1, \ldots, x_i, \ldots$ be a n_e-bounded positive presentation of some concept G_T, where the examples and target are both in k-CORECLASSIC. The incremental learning algorithm LCSLEARN processes the examples x_i one at a time and after each outputs a hypothesis G_{H_i}, resulting in the series of hypotheses $G_{H_0}, \ldots, G_{H_i}, \ldots$ where $G_{H_0} = \emptyset$ and $G_{H_{i+1}} = \text{LCS}(G_{H_i}, x_{i+1})$.

This incremental algorithm will be used to show that k-CORECLASSIC $\cup \{\emptyset\}$ is mb-identifiable with one-sided error (Section 2.3):

THEOREM 5 *k-CORECLASSIC $\cup \{\emptyset\}$ is mb-identifiable with one-sided error, with a mistake bound of ${n_e}^{k+1} + 3{n_e}^k$, regardless of the size of the primitive and role alphabets.*

Proof: Consider a target concept G_T of size n_t, a n_e-bounded positive presentation of G_T $x_1, \ldots, x_i, \ldots$, where the examples and target are both in k-CORECLASSIC, and a series of hypotheses $G_{H_0}, \ldots, G_{H_i}, \ldots$ output by LCSLEARN. Notice that if $x_{i+1} \not\Longrightarrow G_{H_i}$, then $G_{H_{i+1}} \neq G_{H_i}$. From this it follows that after every prediction error

the hypothesis will change; thus a mistake bound can be found indirectly by bounding the number of times that LCSLEARN's hypothesis can change.

First, some terminology: if v_i is a vertex in G_{H_i}, then we will call a vertex of the form (v_i, w) in $G_{H_{i+1}}$ a *successor* of v_i. A vertex v_j is a *descendant* of v_i if it is a successor, or if it is the descendant of some successor of v_i. We now define three types of changes which can take place in computing $G_{H_{i+1}}$. If a vertex v_i in G_{H_i} has two or more successors in $G_{H_{i+1}}$, then we say that a *vertex split* has occurred. If a vertex v_i in G_{H_i} has zero successors in $G_{H_{i+1}}$, then we say that a *vertex drop* has occurred. Finally, if a vertex v_i in G_{H_i} has any successors v_{i+1} in $G_{H_{i+1}}$, such that $\ell_{V_{i+1}}(v_{i+1})$ is a proper subset of $\ell_{V_i}(v_i)$, then we say a *vertex generalization* has occurred. We will use analogous terminology to account for changes in edge labeling: the *successor* of the edge $(v_1, w_1, \mathtt{r})$ in G_{H_i} is the edge $((v_1, v_2), (w_1, w_2), \mathtt{r})$ in $G_{H_{i+1}}$, and edge splits and edge drops are defined by analogy to vertex splits, drops and changes. Notice that if none of these changes occurs then $G_{H_{i+1}}$ is isomorphic to G_{H_i}.

We note first that at most ${n_e}^k$ vertex splits occur in processing the x_i's: by the argument of Lemma 1, the total number of descendants of all vertices in G_{H_1} is ${n_e}^k$, since for every distinct descendant there must be a distinct path in x_1, which has at most ${n_e}^k$ total paths. From this it follows immediately that there are at most ${n_e}^{k+1}$ vertex generalizations (since each vertex v can be generalized at most $|\ell_V(v)|$ times, and $|\ell_V(v)| \leq n_e$) and at most ${n_e}^k$ vertex drops (since each vertex can be dropped at most once.) Also notice that whenever an edge drop occurs, $G_{H_{i+1}}$ has at least one fewer path than G_{H_i}, so there are at most ${n_e}^k$ edge drops. Finally, an edge split only occurs when a vertex split occurs, so they do not need to be counted separately. This leads to the worst case mistake bound given in the theorem. ■

An immediate corollary of the above is that if $n_e \geq n_t$, then the learner must converge to a description equivalent to the target concept G_T after making at most ${n_e}^{k+1} + 3{n_e}^k$ mistakes. The theorem above thus also implies that k-CORECLASSIC is pac-learnable.

While k-CORECLASSIC is a very restricted language, we note that the LCS algorithm of Theorem 4 can be easily extended to many supersets of CORECLASSIC; in particular, additional edge labels and vertex labels, representing additional restrictions on concept membership, can be added to the graph, as long as the range of each labeling function is an upper semi-lattice. The learnability of more expressive subsets of CLASSIC using LCS-like algorithms is discussed at length in another paper (Cohen & Hirsh, 1992).

To summarize, this section has presented a learning algorithm that is easy to implement (it involves nothing much more complicated that DFA intersection), yet whose use yields very strong guarantees of learnability. In particular, not only does the LCS algorithm guarantee pac-learnability, Theorem 5 shows that this algorithm has the much stronger guarantee of one-sided mistake-bounded identifiability, without requiring any assumptions about the size of the primitive and role alphabets used in concepts (as would be necessary using a more traditional approach based on computing the language's VC-dimension).

6. Related work

6.1. *LGG learning algorithms for first-order languages*

Since our learning algorithm is based on computing least common subsumers, one area of related work is the series of studies that have been made on computing the least general generalization (lgg) in logic programming languages. It has long been known that the lgg of two first-order terms can be tractably computed in the lattice of generality imposed by the partial ordering of "θ-subsumption" (Plotkin, 1969). By arguments similar to those used in this paper, it is possible to show that the language of concepts defined by a single atom is efficiently learnable; however, this result has little practical importance because the language is so constrained. Buntine (1988) has extended this algorithm, describing a procedure for computing the "relative lgg" of a set of first-order terms—*i.e.*, the lgg in the lattice imposed by the partial order of "generalized subsumption" in the presence of a background theory. Unfortunately this relative lgg procedure is in general undecidable. Frisch and Page (1990) considered a more restricted version of Buntine's "generalized subsumption" in which the only background information is a taxonomy, and showed that this problem was decidable but NP-hard.

More recent work in this direction has described other first-order languages that can be tractably learned by lgg methods. Džeroski, Muggleton, and Russell (1992) define the class of "constant depth determinate logic programs"[12] and show that this language is learnable, under certain probability distributions, from positive and negative examples. The results of Džeroski, Muggleton, and Russell (which were obtained concurrently with the results of this paper) are broadly similar to ours: the learning algorithm, for example, is also based on the lgg operation. An interesting question is the exact relationship between the language of constant depth determinate Horn clauses and k-CORECLASSIC.

As we argued in Section 3.2, there are difficulties in using a Horn theory (in particular, a constant depth determinate Horn theory) to express concepts like (**ALL** r q) that make use of universal quantification over the fillers of a role; hence k-CORECLASSIC seems to be more expressive in this respect. However, even if only descriptions formed from attributes are considered, there are significant differences between the two languages. For example, given attributes $a_1, \ldots a_n$, the description (**ALL** (a_1 ... a_n) q) is in k-CORECLASSIC for $k = 1$, and hence is easily learnable; however, it corresponds to the determinate clause

$$p(X) \leftarrow a_1(X, Y_1), a_2(Y_1, Y_2), \ldots, a_n(Y_{n-1}, Y_n), q(Y_n)$$

which, since the "depth" of this clause is large, cannot be efficiently learned by the method described in Džeroski, Muggleton, and Russell. Likewise, descriptions of the form (**SAME-AS** (a_1 a_2 ... a_n) (b_1 b_2 ... b_n)) are contained in k-CORECLASSIC; again, however, the corresponding clause

$$p(X) \leftarrow a_1(X, X_1), \ldots, a_n(X_{n-1}, Z), b_1(X, Y_1), \ldots, b_n(Y_{n-1}, Z)$$

is not of constant depth.

To summarize, k-CORECLASSIC appears to be in several ways more expressive than the language of constant depth determinate clauses. The converse is also true, as constant depth determinate logic programs allow predicates of arity greater than two, which are not allowed in CORECLASSIC. Hence the two languages are incomparable.

Our positive result also differs from the result of Džeroski, Muggleton, and Russell in several technical respects. First, our result shows CORECLASSIC to be learnable in a mistake-bounded model from positive examples only, which is stronger than the pac-model used by Džeroski, Muggleton, and Russell. Second, in our result, the learning rate is independent of the size of the vocabulary used to describe examples (*i.e.*, the size of the role and primitive alphabets) whereas in the Džeroski, Muggleton, and Russell result the learning rate is a polynomial function of vocabulary size. Third, Džeroski, Muggleton, and Russell consider the learnability of logic programs containing a constant number of clauses against any "simple" distribution. This problem corresponds roughly to learning the disjunction of a constant number of k-CORECLASSIC descriptions, a problem we have not considered here.

An earlier result describing a first-order language that can be tractably learned using lgg is due to Frisch and Page (1991). They describe a representation called "constrained atoms" which is learnable from positive examples in a mistake-bounded model similar to the model of mb-identifiability presented here. However, while their learning model is more similar to the one used in our positive result, their representation is less similar, as it also allows function symbols (whereas Džeroski, Muggleton, and Russell do not). There is no obvious way of representing function symbols in CORECLASSIC, and hence it seems likely that the language of constrained atoms is not a strict subset of k-CORECLASSIC. However, if function symbols are disallowed, and consistent assumptions are made about the background theory,[13] then the Džeroski, Muggleton, and Russell result strictly generalizes the Frisch and Page result, and thus by the argument above, the languages of k-CORECLASSIC and constrained atoms are incomparable.

6.2. *Negative learnability results*

While formal results showing that the lgg operation is tractable often lead to positive learnability results, the converse does not necessarily hold: since lgg-based methods are only one possible learning algorithm, it is possible for a language to be learnable even if the lgg for that language cannot be tractably computed. Thus results on the intractability of lgg (Buntine, 1988; Frisch & Page, 1990) do not lead immediately to negative results on learnability.

However, some negative results on the learnability of first-order representations do exist. One previous result is due to Haussler (1989), who showed that learning "existentially quantified conjunctive descriptions" without membership queries is NP-hard, even under severe syntactic restrictions; this language is closely related to the language of Horn clauses. The negative results given in Section 4 are for k-CORECLASSIC, a very different first-order language that imposes different restrictions. The complexities in learning CORECLASSIC thus arise for quite different reasons: the culprit here is not existential quantification, as in Haussler's language, but the **SAME-AS** construct, which

allows (very roughly) the equivalent of variable co-reference in first-order logic. Since first-order logic is so powerful, it is not surprising that there are several independent reasons why it is difficult to learn. One of the contributions of this paper—and in particular, of our negative results—is a more complete understanding of the reasons for intractability in learning first-order logics.

Very recently, a number of additional negative results have been obtained that complement the positive result of Džeroski, Muggleton, and Russell. Kietz (1993) shows that a single clause is *not* pac-learnable if the constant-depth determinacy condition does not hold; specifically, it is shown that neither the language of indeterminate clauses of fixed depth nor the language of determinate clauses of arbitrary depth is pac-learnable. These results are dependent on the assumption that the learner's hypothesis is a single clause. Similar results have also been obtained which require cryptographic assumptions, but which are independent of assumptions about the hypotheses of the learning system. Specifically, it has been also shown that determinate clauses of log depth are not polynomially predictable, that recursive determinate clauses of constant depth are not polynomially predictable, and that indeterminate clauses with k "free" variables are polynomially predictable if and only if DNF is polynomially predictable (Cohen, 1993a).

Finally, there have been many analyses of specific first-order learning algorithms in other learnability models; typically, the models either allow queries or do not require polynomially fast convergence. The work presented here has instead focused on the pac-learnability of first-order languages without the use of queries.

7. Concluding remarks

Learning first-order representations is an active area of research in experimental machine learning. In this paper, we have used techniques from computational learning theory to analyze the pac-learnability of certain first-order representations from examples alone. In particular, we have considered the learnability of the restricted first-order logics known as *description logics* (DLs). DLs, sometimes also called "terminological logics" or "KL-ONE-type languages" are subsets of predicate calculus, but are expressed using a nonstandard syntax, allowing a different set of syntactic restrictions to be explored. In contrast to the results of this paper, previous theoretical results on learning first-order concepts have either considered representations based on Prolog rather than DLs (Haussler, 1989; Frisch & Page, 1991; Džeroski et al., 1992) or have used weaker models of learnability (Shapiro, 1982; Muggleton & Buntine, 1988).

After describing the DL CORECLASSIC, we presented two main learnability results. The first result showed that the full language is not learnable, by exploiting the expressive power allowed by the **SAME-AS** construct, which (roughly) allows the equivalent of variable co-reference in first-order logic. We then demonstrated the existence of a hierarchy of learnable subsets of CORECLASSIC obtained by syntactically restricting the language. In particular, for any constant k, one can tractably learn the the class of k-CORECLASSIC descriptions: descriptions that have either acyclic concept graphs of depth bounded by k, or acyclic concept graphs formed from at most k **SAME-AS** constructs. This language appears to be incomparable in expressive power to any subset of

first-order logic previously known to be learnable. Interestingly, these learnability results hold even if the alphabets of primitive classes and roles are infinite; our positive result thus generalizes not only the result of Valiant (1984) on learning monomials to learning concepts in our (conjunctive) first-order language, but also the result of Blum (1990) on learning monomials over infinite attribute spaces.

In future work, we seek to extend these results by considering the learnability of more expressive DLs; one initial effort in this direction has been to extend these results to the full CLASSIC DL (Cohen & Hirsh, 1992). A prototype learning system based on these results has also been implemented, although our experimentation with it is so far at an extremely early stage.

A more detailed comparison of learnable DLs to the learnable subsets of Prolog is also an interesting formal problem, and one which we hope to address in the future. The main obstacle in making a such comparison at this date is that it will require a precise formal understanding of the boundaries of learnability for Horn clause logic, and these boundaries are only now beginning to be understood (Cohen, 1993b; Cohen 1993a; Kietz, 1993).

Acknowledgments

This research was conducted while Haym Hirsh was consulting at AT&T Bell Laboratories. The authors would like to thank Rob Schapire and Mike Kearns for comments on a draft of the paper, and Alex Borgida for comments on early version of some of the proofs. We would also like to thank Lenny Pitt for finding a technical error in an earlier version of the paper.

Notes

1. This follows previous work on learning finite automata (Angluin, 1987). We will assume that n_e is not known to the learning algorithms.
2. An additional motivation for us is that in many implemented DLs, it is possible to attach an arbitrary description to an instance (for example, it may be possible to record about an instance "Doris" the universally-quantified assertion that all of her daughters are doctors), and hence the distinction between instances and concepts is blurred.
3. For expository purposes we will ignore the distinction between syntax and semantics and use the same notation for a CORECLASSIC symbol, the set which corresponds to it, and the first-order predicate corresponding to that set.
4. That is, $\forall x, y, z,\ \ \mathtt{r}(x, y) \wedge \mathtt{r}(x, z) \ \Rightarrow\ y = z$.
5. The relations $\mathtt{r}$ and $\mathtt{q}$ are provided by the user, just as the user of CORECLASSIC must provide definitions of the relations corresponding to primitive concepts and roles.
6. "Basic CLASSIC" is closely related to CORECLASSIC; the differences are that it includes a few more description constructors, allows for a hierarchy of primitive concepts, and restricts the attribute chains in **SAME-AS** clauses to be of nonzero length.
7. We note that CORECLASSIC concept graphs differ from the concept graphs used by Haussler (1989) in two important respects: Haussler's concept graphs are not rooted, and arcs are interpreted as existentially (rather than universally) quantified restrictions. Because of these differences the properties of CORECLASSIC graphs with respect to learnability are quite different.

8. For technical reasons, our hardness result requires the assumption that $RP \neq NP$, rather than the more usual assumption that $P \neq NP$. RP is the class of problems that can be solved in polynomial time with high probability by a randomized algorithm; it would be almost as surprising to find out that $RP = NP$ as to find out that $P = NP$.
9. Our definition and notation for DFAs follow that of Hopcroft and Ullman (1979).
10. Up to equivalence under $\equiv$, where $G_1 \equiv G_2$ iff $G_2 \Longrightarrow G_1$ and $G_1 \Longrightarrow G_2$.
11. We omit a proof of this fact; note however that by the construction of Theorem 1, the concept graph of such a description can have at most k vertices of indegree greater than 1.
12. To briefly summarize the definition of this class, let $A \leftarrow B_1 \wedge \ldots \wedge B_r$ be an (ordered) Horn clause and let DB be a background theory. The *input variables* of the literal B_i are those variables appearing in B_i which also appear in the clause $A \leftarrow B_1 \wedge \ldots \wedge B_{i-1}$; all other variables appearing in B_i are called *output variables*. A literal is determinate if its output variables have only one possible binding, given DB and the binding of the input variables, and a clause is determinate if all of its literals are determinate. Next, define the *depth* of a variable appearing in a clause $A \leftarrow B_1 \wedge \ldots \wedge B_r$ as follows. Variables appearing in the head of a clause have depth zero. Otherwise, let B_i be the first literal containing the variable V, and let d be the maximal depth of the input variables of B_i; then the depth of V is $d + 1$. The depth of a clause is the maximal depth of any variable in the clause. Džeroski, Muggleton, and Russell consider the language of determinate clauses of depth less than i over a background theory DB that contains only ground atomic clauses of arity bounded by j, where i and j are constants.
13. Frisch and Page use a "constraint language" that fills the same role as the background theory used by Džeroski, Muggleton, and Russell; however, while Džeroski, Muggleton, and Russell assume the background theory to be a set of ground unit clauses, Frisch and Page assume that questions to the constraint language are answered by an oracle.

References

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

Banerji, R. (1988). Learning theories in a subset of polyadic logic. In *Proceedings of the 1988 Workshop on Computational Learning Theory* (pp. 267-278), Boston, Massachusetts.

Beck, H. (1991). Language acquisition from cases. In *Proceedings of the 1991 DARPA Case-Based Reasoning Workshop* (pp. 159-169), Washington, D.C.

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

Blum, A. (1990). Learning boolean functions in a infinite attribute space. In *22nd Annual Symposium on the Theory of Computing* (pp. 373-386). ACM Press.

Blumer, A., Ehrenfeucht, A., Haussler, D., & Warmuth, M. (1989). Classifying learnable concepts with the Vapnik-Chervonenkis dimension. *Journal of the Association for Computing Machinery*, 36(4):929–965.

Bobrow, D. & 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. Technical Report DCS-TR-295, Rutgers University Department of Computer Science.

Borgida, A. (1993). On the relationship between description logics and predicate logic queries. Technical Report DCS-TR-295a, Rutgers University Department of Computer Science.

Borgida, A. & Patel-Schneider, P. F. (1994). A semantics and complete algorithm for subsumption in the CLASSIC description logic. *Journal of Artificial Intelligence Research*, 1:277–308.

Brachman, R. J. (1979). On the epistomological status of semantic networks. In Findler, N. V., editor, *Associative networks: representation and use of knowledge by computers*. Academic Press.

Buntine, W. (1988). Generalized subsumption and its application to induction and redundancy. *Artificial Intelligence*, 36(2):149–176.

Cohen, W. & Hirsh, H. (1992). Learnability of the CLASSIC 1.0 knowledge representation language. AT&T Bell Labs Technical Memorandum. Available from the author on request.

Cohen, W. W. (1993a). Cryptographic limitations on learning one-clause logic programs. In *Proceedings of the Tenth National Conference on Artificial Intelligence* (pp. 80-85), Washington, D.C.

Cohen, W. W. (1993b). A pac-learning algorithm for a restricted class of recursive logic programs. In *Proceedings of the Tenth National Conference on Artificial Intelligence* (pp. 86-92), Washington, D.C.

Cohen, W. W., Borgida, A., & Hirsh, H. (1992). Computing least common subsumers in description logics. In *Proceedings of the Tenth National Conference on Artificial Intelligence* (pp. 754-760), San Jose, California. MIT Press.

Conklin, D. & Gasglow, J. (1992). Spatial analogy and subsumption. In *Proceedings of the Ninth International Conference on Machine Learning* (pp. 111-116), Aberdeen, Scotland. Morgan Kaufmann.

Devanbu, P., Brachman, R. J., Selfridge, P., & Ballard, B. (1991). LaSSIE: A knowledge-based software information system. *Communications of the ACM*, 34(5), 34-49.

Dietterich, T. & Michalski, R. (1983). A comparative review of selected methods for learning from examples. In *Machine Learning: An Artificial Intelligence Approach*. Morgan Kaufmann.

Dietterich, T. G., London, B., Clarkson, K., & Dromey, G. (1982). Learning and inductive inference. In Cohen, P. and Feigenbaum, E. A., editors, *The Handbook of Artificial Intelligence, Volume III*. William Kaufmann, Los Altos, CA.

Džeroski, S., Muggleton, S., & Russell, S. (1992). Pac-learnability of determinate logic programs. In *Proceedings of the 1992 Workshop on Computational Learning Theory* (pp. 128-135), Pittsburgh, Pennsylvania.

Frisch, A. & Page, C. D. (1990). Generalization with taxonomic information. In *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 755-761), Boston, Massachusetts. MIT Press.

Frisch, A. & Page, C. D. (1991). Learning constrained atoms. In *Proceedings of the Eighth International Workshop on Machine Learning* (pp. 427-431), Ithaca, New York. Morgan Kaufmann.

Gold, M. (1967). Language identification in the limit. *Information and Control*, 10, 447-474.

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

Helmbold, D., Sloan, R., & Warmuth, M. (1990). Learning nested differences of intersection-closed concept classes. *Machine Learning*, 5(2), 165-196.

Hirsh, H. (1990). *Incremental Version Space Merging: A General Framework for Concept Learning*. Kluwer Academic Publishers.

Hopcroft, J. E. & Ullman, J. D. (1979). *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley.

Idestam-Almquist, P. (1993). Generalization under implication by recursive anti-unification. In *Proceedings of the Ninth International Conference on Machine Learning* (pp. 151-158), Amherst, Massachusetts. Morgan Kaufmann.

Kearns, M., Li, M., Pitt, L., & Valiant, L. (1987). Recent results in boolean concept learning. In *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 337-352), Ithaca, New York. Morgan Kaufmann.

Kearns, M. & Valiant, L. (1989). Cryptographic limitations on learning Boolean formulae and finite automata. In *21th Annual Symposium on the Theory of Computing* (pp. 433-444). ACM Press.

Kietz, J.-U. (1993). Some computational lower bounds for the computational complexity of inductive logic programming. In *Proceedings of the 1993 European Conference on Machine Learning* (pp. 115-123), Vienna, Austria.

Littlestone, N. (1988). Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4), 285-318.

MacGregor, R. M. (1991). The evolving technology of classification-based knowledge representation systems. In Sowa, J., editor, *Principles of semantic networks: explorations in the representation of knowledge*. Morgan Kaufmann.

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

Minsky, M. (1975). A framework for representing knowledge. In Winston, P., editor, *The psychology of computer vision*. McGraw-Hill.

Morik, K. (1989). A bootstrapping approach to conceptual clustering. In *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 503-504), Ithaca, New York. Morgan Kaufmann.

Muggleton, S. & Buntine, W. (1988). Machine invention of first order predicates by inverting resolution. In *Proceedings of the Fifth International Conference on Machine Learning* (pp. 339-352), Ann Arbor, Michigan. Morgan Kaufmann.

Muggleton, S. & Feng, C. (1992). Efficient induction of logic programs. In *Inductive Logic Programming*. Academic Press.

Muggleton, S. H., editor, (1992). *Inductive Logic Programming*. Academic Press.

Natarajan, B. K. (1987). On learning boolean functions. In *19th Annual Symposium on the Theory of Computing* (pp. 296-352). ACM Press.

Pfenning, F. (1991). Unification and anti-unification in the calculus of constructions. In *Sixth Annual IEEE Symposium on Logic in Computer Science* (pp. 74-85). Amsterdam: The Netherlands. Also available as Ergo Report 91-096, School of Computer Science, CMU, Pittsburgh.

Pitt, L. & Valiant, L. (1988). Computational limitations on learning from examples. *Journal of the ACM*, 35(4):965–984.

Pitt, L. & Warmuth, M. (1990). Prediction-preserving reducibility. *Journal of Computer and System Sciences*, 41:430–467.

Plotkin, G. D. (1969). A note on inductive generalization. *Machine Intelligence, 5*, 153–163.

Quillian, M. R. (1967). Word concepts: a theory and simulation of some basic semantic capabilities. *Behavioral Science*, 12:410–430.

Quinlan, J. R. (1990). Learning logical definitions from relations. *Machine Learning*, 5(3), 239-266.

Rivest, R. L. (1987). Learning decision lists. *Machine Learning*, 2(3), 229-246.

Schewe, K. D. (1989). Variant construction using constraint propagation techniques over semantic networks. In *Proceedings of the 5th Austrian AI Conference*, Insbruck.

Shapiro, E. (1982). *Algorithmic Program Debugging*. MIT Press.

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

Vilain, M., Koton, P., & Chase, M. (1990). On analytical and similarity-based classification. In *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 867-874), Boston, Massachusetts. MIT Press.

Winston, P. (1975). Learning structural descriptions from examples. In Winston, P., editor, *The psychology of computer vision*, pages 157–209. McGraw-Hill.

Woods, W. A. & Schmolze, J. G. (1992). The KL-ONE family. *Computers and Mathematics with Applications*, 23(2-5).

Received April 26, 1993
Accepted September 20, 1993
Final Manuscript April 14, 1994