Logical settings for concept-learning

Luc De Raedt
Department of Computer Science, Katholieke Universiteit Leuven
Celestijnenlaan 200A, B-3001 Heverlee, Belgium
email: Luc.DeRaedt@cs.kuleuven.ac.be
Tel: ++ 32 16 32 76 43 Fax : ++ 32 16 32 79 96

June 16, 1997

Abstract

Three different formalizations of concept-learning in logic (as well as some variants) are analyzed and related. It is shown that learning from interpretations reduces to learning from entailment, which in turn reduces to learning from satisfiability. The implications of this result for inductive logic programming and computational learning theory are then discussed, and guidelines for choosing a problem-setting are formulated.

Keywords: inductive logic programming, computational learning theory, concept-learning, logic.

1 Introduction

Various formalisations of concept-learning in logic have been proposed, e.g. learning from interpretations [Valiant, 1984; De Raedt and Džeroski, 1994; Angluin *et al.*, 1992], learning from entailment [Frazier and Pitt, 1993], and several inductive logic programming settings [Muggleton and De Raedt, 1994; De Raedt, 1996; Flach, 1992; Flach, 1995; Wrobel and Džeroski, 1995]. These formalizations differ in their representation of examples and the corresponding membership function or coverage notion, which determines whether an example is covered by a concept. At present, it is an open question as to what the relation among these different formalizations is. This question has been raised in different forms and in different contexts. Firstly, in computational learning theory, [Angluin, 1987; Frazier and Pitt, 1993; Angluin *et al.*, 1992] have wondered what the general relation is between learning from interpretations and learning from entailment, and which setting should be used for which type of learning problem. Secondly, in inductive logic programming it is unclear how systems such as FOIL [Quinlan, 1990] and Golem [Muggleton and Feng, 1990], which learn from entailment, relate to systems such as ICL [De Raedt and Van Laer, 1995] and Claudien [De Raedt and Dehaspe, 1997], which learn from interpretations. Furthermore, several variants have been proposed of these settings [Fensel *et al.*, 1995;

Wrobel and Džeroski, 1995]. Thirdly, though computational learning theory and attribute-value learning techniques typically learn from interpretations, inductive logic programming has mainly addressed learning from entailment, which indicates that answers to the open question may also increase our understanding of the relation between inductive logic programming and attribute value learning. Fourthly, PAC-learning studies of inductive logic programming such as [Cohen and Page, 1995; De Raedt and Džeroski, 1994] have also left open the question as to whether learning from interpretations allows to PAC-learn classes of concepts that are not learnable from entailment.

This paper contributes answers to the open questions raised above. More specifically, it will be shown that learning from interpretations reduces to learning from entailment, which in turn reduces to learning from satisfiability. Learning from satisfiability is a more recent setting that generalizes the other settings. In this setting, examples and hypotheses are both full clausal theories.

It will also be shown that there exist propositional classes such as k-CNF that are PAC-learnable from interpretations, but that are not efficiently learnable from entailment or from satisfiability. The implications of this result for inductive logic programming, computational learning theory and attribute value learning are then analyzed. Furthermore, practical suggestions for choosing the right problem-setting are formulated.

The paper is organised as follows : section 2 briefly reviews some basic notions in logic; section 3 introduces the different formalisations of concept-learning in clausal logic and investigates the relation among them; section 4 discusses the implications of this relation for computational learning theory (this section may be skipped by the casual reader less interested in theory); section 5 touches upon related work, and finally, section 6 concludes.

2 Logic

We first review some standard concepts from the predicate calculus (see e.g. [Genesereth and Nilsson, 1987] for more details).

In this paper, an ***alphabet*** consists of a set of constant, functor and predicate symbols.

A ***term*** t is either a constant, a variable or a compound term $f(t_1, ..., t_n)$ composed of an n-ary function symbol f, and n terms t_i. An ***atom*** is a logical formula of the form $p(t_1, .., t_n)$, where p is an n-ary predicate symbol and t_i are terms. A ***literal*** is an atom or the negation $\neg A$ of an atom A. Atoms are positive literals, and negated atoms are negative literals.

In propositional logic, no terms are considered. Thus in propositional logic, all predicates have 0 arity. In computational learning, one often employs CNF formulae, which are of the following form:

$$(l_{1,1} \vee ... \vee l_{1,n_1}) \wedge ... \wedge (l_{k,1} \vee ... \vee l_{k,n_k})$$

where all $l_{i,j}$ are propositional literals. CNF expressions have also a natural upgrade in first order logic, namely clausal theories (or CT, for short):

$$(\forall V_{1,1}, ..., V_{1,v_1} : l_{1,1} \vee ... \vee l_{1,n_1}) \wedge ... \wedge (\forall V_{k,1}, ..., V_{1,v_k} : l_{k,1} \vee ... \vee l_{k,n_k})$$

where all $l_{i,j}$ are literals and $V_{i,1}, ..., V_{i,v_i}$ are all variables occurring in $l_{i,1} \vee ... \vee l_{i,n_i}$. The symbol $\forall$ reads ***for all*** and stands for universal quantification. Each disjunction is called a

clause. A clause $(\forall V_1, ..., V_v : h_1 \vee ... \vee h_n \vee \neg b_1 \vee ... \vee \neg b_m)$ is often written as an implication $h_1 \vee ... \vee h_n \leftarrow b_1 \wedge ... \wedge b_m$.

Interesting subsets of clausal logic, are ***Horn*** (resp. ***definite***) clause logic. They consist of CT expressions that have at most one (resp. exactly one) positive literal in each clause.

Interpretations are used to formalize truth and falsity of specific formulae and entailment. As we use only clausal logic, we will focuss on so-called ***Herbrand*** interpretations, which can – for our purposes – be defined as sets of variable-free (i.e. ***ground***) atoms over a given alphabet. According to a Herbrand interpretation all atoms in the interpretation are true, and all other atoms (over the alphabet) are false. In the propositional or boolean case, a Herbrand interpretation corresponds to a variable assignment, i.e. a truth assignment to the propositional atoms which are the 'variables' of the formula.

A substitution $\theta = \{V_1 \leftarrow t_1, ..., V_n \leftarrow t_n\}$ is an assignment of terms $t_1, ..., t_n$ to variables $V_1, ..., V_n$. The formula $F\theta$ where F is a term, atom, literal or expression, and $\theta = \{V_1 \leftarrow t_1, ..., V_n \leftarrow t_n\}$ is a substitution, is the formula obtained by simultaneously replacing all variables $V_1, ..., V_n$ in F by the terms $t_1, ..., t_n$.

A clausal theory T is true in a Herbrand interpretation I if $T\theta$ is true in I for each substitution θ for which $T\theta$ is ground. A ground clausal theory $T\theta$ is true in I if and only if each clause of $T\theta$ is true in I. A ground clause is true in I if one of the atoms that appears positively in the clause is true according to I or one of the atoms that appears negatively is false according to I. E.g. $flies \vee \neg bird \vee \neg abnormal$ is true in the interpretations $\{flies\}, \{abnormal\}$ but false in $\{bird, abnormal\}$. If a theory is true in an interpretation we also say that the interpretation is a ***model*** for the theory.

Logical entailment and satisfiability are typically defined using interpretations. We will write $F \models G$ (read ***F logically entails G***) when all models of F are also a model for G, and $F \models \square$ (read ***F is not satisfiable***) if there exists no interpretation that is a model of F.

3 Concept-learning

In concept-learning [Mitchell, 1982], one is given a language of concepts L_C, a language of examples L_e, the ***covers*** or membership-relation $\in_C$ that specifies how L_C relates to L_e, and a set of examples E of an unknown target concept $t \in L_C$. Each example is of the form $(e, Class)$ where $e \in L_e$ and $Class$ is ***true*** or ***false***. Examples $(e, true)$ are positive examples, whereas examples $(e, false)$ are negative. The aim in concept-learning is then to find a hypothesis $H \in L_C$ that covers all positive examples (i.e. H is complete) and none of the negative examples (i.e. H is consistent).

3.1 Concept-learning in logic

Logical approaches to concept-learning instantiate this definition by defining the representation languages for concepts L_C and examples L_e, as well as the coverage $\in_C$ relation among them. We will now investigate several possibilities for formalizing concept-learning in clausal logic.

Roughly speaking, in inductive logic programming, hypotheses are clausal theories, examples are clauses, and a hypothesis covers an example if the hypothesis logically entails the example (cf. also section 3.2.):

Definition 1 *(learning from entailment) If H is a clausal theory and e a clause, then H covers e under entailment, notation $e \in_e H$, if and only if $H \models e$.*

We will call this setting ***learning from entailment*** following [Frazier and Pitt, 1993].

Example 1 *Let the examples be ($flies \leftarrow normal \wedge bird$; **true**), ($flies \leftarrow bird; false$) and ($\leftarrow flies \wedge normal \wedge bird$; **false**). Then $flies \leftarrow bird \wedge normal$ is a solution.*

Another logical setting originates from the work on PAC-learning [Valiant, 1984; De Raedt and Džeroski, 1994]. In ***learning from interpretations***, hypotheses are clausal theories, examples are Herbrand interpretations and an example is covered when it is a model for the hypothesis.

Definition 2 *(learning from interpretations) If H is a clausal theory and e a Herbrand interpretation, then H covers e under interpretations, notation $e \in_i H$, if and only if e is a model for H.*

Example 2 *Let examples be ($\{bird, normal, flies\}$;**true**) and ($\{normal, bird\}$; **false**). Then $flies \leftarrow bird \wedge normal$ is a solution.*

When learning from interpretations, it is implicitly assumed that each example is completely specified. Indeed, in propositional logic, all propositions should be either true or false. As a consequence, missing values cannot be represented in this framework. It has therefore been suggested by [Fensel ***et al.***, 1995] to represent examples by partial interpretations. In a partial interpretation, certain ground atoms have an unknown truth-value (see below for a formal definition). Alternatively, [Flach, 1992] employs a second-order logic for dealing with this situation.

A generalization of learning from partial interpretations, called ***learning from satisfiability***, is defined below.

Definition 3 *(learning from satisfiability) If H and e are both clausal theories, then H covers e under satisfiability, notation $e \in_s H$, if and only if $H \wedge e \not\models \Box$.*

Example 3 *Let the examples be ($\{bird \leftarrow; normal \leftarrow; flies \leftarrow\}$; **true**), ($\{bird \vee flies \leftarrow; normal \leftarrow\}$; **true**) and ($\{bird \leftarrow; normal \leftarrow; \leftarrow flies\}$; **false**). Then $flies \leftarrow bird \wedge normal$ is a solution.*

This notion of coverage (using the membership function $\in_s$) was proposed by [Wrobel and Džeroski, 1995]. However, [Wrobel and Džeroski, 1995] did not choose full clausal logic to represent examples and hypotheses. Yet, this choice is essential for our purposes.

Whereas learning from entailment and learning from interpretations are well-known and well-motivated in the literature, we still need to answer the question as to ***Why learning from satisfiability is useful ?*** A first and tentative answer is that learning from satisfiability seems the most general setting possible within clausal logic as both examples and hypotheses are clausal theories. We will soon show that the other settings defined above indeed reduce to learning from satisfiability.

3.2 Relation among the different settings

We will now investigate the relation between the different formalizations of concept-learning in logic using the concept of a solution-set:

Definition 4 $sol_x(E) = \{H \mid H$ *is a clausal theory over a given alphabet such that for all* $(p, true) \in E : p \in_x H$ *and for all* $(n, false) \in E : n \notin_x H\}$

To investigate how one type of coverage notion relates to another type of coverage, we use reductions[1]:

Definition 5 *A reduction from learning under* $\in_x$ *to learning under* $\in_y$ *is a function* ρ *that maps any example set* E *(under* $\in_x$*) onto an example set* $E_\rho = \{\rho(e) \mid e \in E\}$ *(under* $\in_y$*) such that* $sol_x(E) = sol_y(E_\rho)$.

If there exists a reduction ρ from learning under $\in_x$ to learning under $\in_y$, we can solve learning problems under $\in_x$ using the algorithms for $\in_y$. One merely has to map the example set E to E_ρ and run the algorithm under $\in_y$. The solutions generated under $\in_y$ will also be solutions under $\in_x$. We can then consider learning under $\in_y$ a harder or more general task than learning under $\in_x$. Obviously:

Property 1 *If there exist reductions from learning under* $\in_x$ *to learning under* $\in_y$ *and from learning under* $\in_y$ *to learning under* $\in_z$*, then there exists a reduction learning under* $\in_x$ *to learning under* $\in_z$.

Some of the reductions will represent Herbrand interpretations by clausal theories as follows[2]:

Definition 6 *Let* i *be a Herbrand interpretation in which* $t_1, ..., t_n$ *are the true facts, and* $f_1, ..., f_m$ *are the false facts. Then* $\overline{i}$ *denotes the clausal theory* $\{t_1 \leftarrow; ...; t_n \leftarrow; \leftarrow f_1; ...; \leftarrow f_m\}$.

One property of this transformation that follows directly from Proposition 3.2. in [Lloyd, 1987] and that will be used in some of the proofs, goes as follows:

Property 2 *Let* H *be a clausal theory and* i *be a Herbrand interpretation. Then* i *is a model for* H *iff* $\overline{i} \wedge H \not\models \Box$.

Let us now investigate the relation among learning from entailment and learning from satisfiability.

[1]When talking about reductions, it will always be assumed that the language of concepts (including the alphabet) is fixed. This assumption is needed because some of the reductions studied below change the alphabet of the examples (using skolems).

[2]From a practical perspective, the definition can only be applied to finite Herbrand interpretations. With finite interpretations, we mean interpretations whose sets of true and false facts are both finite. However, Property 2 also holds for infinite Herbrand intepretations, which merely result in a theory consisting of an infinite number of clauses. Clausal theories may - in general - contain an infinite number of clauses; clauses must contain a finite number of literals.

Theorem 1 *Learning from entailment reduces to learning from satisfiability.*

Proof: Define $\rho((e, Class)) = (\neg e, \neg Class)$, where e is a clause. The result then follows from the observation that $e \in_e H$ iff $\neg e \notin_s H$. Notice that if $h_1 \vee ... \vee h_n \leftarrow b_1 \wedge ... \wedge b_n$ is a clause, its negation is the clausal theory $\leftarrow h_1\sigma; ...; \leftarrow h_n\sigma; b_1\sigma \leftarrow; ...; b_m\sigma \leftarrow$, where σ is a skolem substitution[3]. $\square$

This theorem shows that ***learning from entailment*** is to be considered a special case of ***learning from satisfiability.*** The converse does not seem to hold, as clausal theories cannot in general be transformed into single clauses.

Consider now the relation among learning from interpretations and learning from entailment:

Theorem 2 *Learning from finite interpretations reduces to learning from entailment.*

Proof: Define $\rho((e, Class)) = (\neg\overline{e}, \neg Class)$, where e is a finite Herbrand interpretation. The result then follows from the observation that $i \in_i H$ iff $\neg\overline{i} \notin_e H$[4], which in turn follows from Property 2. Notice that if i is an interpretation with as true facts $t_1, ..., t_m$ and as false facts $f_1, ..., f_n$, then $\neg\overline{i}$ is the clause $f_1 \vee ... \vee f_n \leftarrow t_1 \wedge ... \wedge t_m$. $\square$.

Corollary 1 *Learning from finite interpretations reduces to learning from satisfiability.*

Notice that the converse of theorem 2 does not hold, as the negation of a clause is not necessarily a (complete) interpretation. It can however be considered a partial interpretation, as some facts will be true, others will be false, and still others will have an unknown truth-value. More formally: a ***partial*** interpretation (over an alphabet) consists of a set of true ground facts T and a set of false ground facts F. A Herbrand interpretation I (over the same alphabet) ***extends*** a partial interpretation (T, F) if and only if $T \subseteq I$ and $F \cap I = \emptyset$.

Definition 7 *(learning from partial interpretations) If H is a clausal theory and e is a partial interpretation then H covers e under partial interpretations, notation $e \in_{pi} H$, if and only if e has an extension I that is a model of H[5].*

As for Herbrand interpretations, $\overline{e}$ denotes the clausal theory corresponding to the partial interpretation e. Furthermore, there is a one-to-one correspondence between finite partial interpretations e and clauses $\neg\overline{e}$. Using this mapping it is easy to prove that :

Theorem 3 *Learning from finite partial interpretations reduces to learning from entailment, and vice versa, learning from entailment reduces to learning from finite partial interpretations.*

[3]A skolem substitution substitutes all variables by different constants that are not in the current alphabet.

[4]This restriction to finite interpretations is needed to guarantee that the resulting expressions $\neg\overline{i}$ are finite clauses. This is not really a strong restriction, as infinite interpretations cannot be represented explicitly, but see [De Raedt and Džeroski, 1994].

[5]A related learning setting is considered by [Greiner ***et al.***, 1996], who require that all extensions are a model of the hypothesis.

This theorem demonstrates that learning from entailment and learning from partial interpretations are essentially equivalent. Hence, we will not further distinguish among them.

Finally, within inductive logic programming one typically employs also a background theory B in the form of a clausal theory, and regards an example e covered by a hypothesis H only if $B \wedge H \models e$.

Definition 8 *(intensional inductive logic programming) If H and B are clausal theories and e is a clause, then H covers e under intensional inductive logic programming, notation $e \in_{int,B} H$, if and only if $B \wedge H \models e$.*

Theorem 4 *Intensional inductive logic programming reduces to learning from satisfiability.*

Proof: Define $\rho((e, Class)) = (B \wedge \neg e, \neg Class)$ and consider that $e \in_{int,B} H$ iff $B \wedge \neg e \notin_s H$. $\Box$

The reduction of ***intensional inductive logic programming*** to ***learning from satisfiability*** forms another motivation for studying the latter setting.

A special case, frequently applied in inductive logic programming (e.g. the well-known systems Golem [Muggleton and Feng, 1990] and Foil [Quinlan, 1990]) and its computational learning theory formalisation, assumes that the background theory B consists of a set of ground atoms and that the positive examples are true ground atoms and the negative ones false ground atoms. This setting is known in the literature as the ***extensional*** inductive logic programming setting.

Definition 9 *(extensional inductive logic programming) If H is a clausal theory, B a set of true ground atoms, and e a ground atom, then H covers e under extensional inductive logic programming, notation $e \in_{ext,B} H$, if and only if $B \wedge H \models e$.*

Theorem 5 *Extensional inductive logic programming reduces to learning from entailment.*

Proof: Define $\rho((e, Class)) = (e \leftarrow B, Class)$. The result then follows from the fact that $e \in_{ext,B} H$ iff $H \models (e \leftarrow B)$. $\Box$

Unfortunately, learning from entailment seems not reducible in this manner to extensional inductive logic programming. This is because the above transformation assumes that when learning from entailment all examples have the same condition part. This assumption does not hold in general. Consider e.g. learning $p \leftarrow q \wedge r$ from the positive example $p \leftarrow q \wedge r \wedge t$ and the negatives $p \leftarrow q$, and $p \leftarrow r$[6].

So far we ignored the fact that many approaches (especially in the domain of inductive logic programming) assume that the examples when learning from entailment are Horn clauses. Let us name this setting ***Horn-learning from entailment***. Trivially, this setting can be reduced to learning from entailment. However, it seems impossible to reduce ***learning from interpretations*** to ***Horn-learning from entailment***. The reason is that the clauses $\neg \overline{e}$ obtained from Herbrand interpretations e, are typically not Horn.

[6]One might want to consider taking as the extensional background theory the union of the antecedents of the examples, and as examples the consequence of the examples. Unfortunately, this does not work (cf. the illustration). To make this approach work, one should also change the representation by adding an extra argument to all of the predicates. This argument would then contain a unique identifier for the example. However, such changes of representation are not permitted within our (strict) notion of reduction.

3.3 Knowledge representation

The three main settings, i.e. learning from interpretations, from partial interpretations[7], and from satisfiability, can also be interpreted from a knowledge representation perspective.

Central to this issue is the question as to ***what an example represents ?*** The question can best be answered in terms of model theory. In terms of model theory, each example e corresponds to a set of models $M(e)$, and an example e is covered by a hypothesis H if and only if there is a model $m \in M(e)$ that is a model of H. Formally:

Definition 10 ***If** e **is a Herbrand interpretation** I **then** $M(e) = \{I\}$**; if** e **is a partial interpretation then** $M(e) = \{I \mid I$ **is a Herbrand extension of** e $\}$**; if** e **is a clausal theory then** $M(e) = \{I \mid I$ **is a Herbrand model for** $e\}$*

Property 3 $e \in_x H$ ***iff*** $\exists m \in M(e)$: ***m is a model for*** H ***where*** $\in_x = \in_i$ ***or*** $\in_{pi}$ ***or*** $\in_s$.

This property suggests that the main difference among the three formalizations of concept-learning is due to the models $M(e)$ that an example represents. When learning from interpretations, $M(e)$ contains a single interpretation. By definition, an interpretation assumes complete knowledge. Hence, when learning from interpretations, complete knowledge about each of the examples is assumed. When learning from partial interpretations, $M(e)$ contains all extensions of the partial interpretations. The difference between the extensions and the partial model is that the extensions assign the value true or false to the facts that have an unknown truth-value in the partial interpretations. Hence, partial interpretations can represent examples with missing values. When learning from clausal theories, $M(e)$ can (depending on the example) contain any set of Herbrand interpretations. E.g. assume that the truth-value of two propositional facts p and q is not known, but it is known that they have identical truth-values. One cannot represent this knowledge using a partial interpretation. However, using the clausal theory $p \leftarrow q$ and $q \leftarrow p$ will realize the desired effect. This example illustrates that learning from satisfiability allows us to express other types of incomplete knowledge.

This knowledge representation view provides guidance for choosing the right setting when modelling an induction task. If complete knowledge about each of the examples is available, use ***learning from interpretations***; if some examples have missing values, use ***learning from partial interpretations*** or ***learning from entailment***; if other forms of incomplete examples need to be represented, use ***learning from satisfiability.***

4 Computational learning theory

The PAC-learnability of several subclasses of clausal logic has been investigated under various membership relations. We first formalize the PAC-learning paradigm introduced by [Valiant, 1984], and then investigate the role of the membership relation for PAC-learning.

[7]Which is considered here to be equivalent to learning from entailment, cf. above.

4.1 PAC-learning: definition

Let L_C be a class of concepts. The target concept t may be any concept in L_C. A learning algorithm for L_C is an algorithm that attempts to construct an approximation to the target concept from examples for it. The learning algorithm takes as input two parameters: the error parameter $\epsilon \in (0,1]$ and the confidence parameter $\delta \in (0,1]$. The error parameter specifies the error allowed in a good approximation and the confidence parameter controls the likelihood of constructing a good approximation.

The learning algorithm has at its disposal a subroutine EXAMPLE, which at each call produces a single example for the target concept t. The probability that a particular example $e \in L_e$ (positive or negative for t) will be produced at a call of EXAMPLE is $D(e)$, where D is an arbitrary unknown but fixed distribution on L_e. The choice of the distribution D is independent of the target concept t.

Concept g is a good approximation of concept t if the probability that f and g differ on a randomly chosen example from L_e is at most ϵ, i.e. $D(t\Delta g) \leq \epsilon$, where $t\Delta g = t - g \cup g - t$. Putting all of the above together, we obtain the following definition.

Definition 11 *An algorithm A is a* probably approximately correct (PAC) *learning algorithm for a class of learning tasks (L_C, L_e, $\in_C$) if*

1. *A takes as input $\epsilon \in (0,1]$ and $\delta \in (0,1]$.*
2. *A calls EXAMPLE, which returns examples for some unknown but fixed $t \in L_C$. The examples are chosen randomly according to an unknown but fixed probability distribution D on L_e.*
3. *For all concepts $t \in L_C$ and all probability distributions D on L_e, A outputs a concept $g \in L_C$, such that with probability at least $(1-\delta)$, $D(t\Delta g) \leq \epsilon$.*
4. *The time complexity of A is bounded by a polynomial $p(1/\epsilon, 1/\delta, m, size(t))$ where m is the size of the largest example, and $size(t)$ the size of the target concept.*

A class L_C is PAC-learnable *under $\in_C$ if there exists an algorithm A which is a PAC-learning algorithm for $(L_C, L_e, \in_C)$.*

Notice that the membership functions $\in_x$ considered in this paper completely determine the language of examples L_e used. Hence, we say L_C is *PAC-learnable under* $\in_C$ instead of L_C is *PAC-learnable under $\in_C$ for the language of examples L_e corresponding to $\in_C$*.

To prove PAC-learning results, one frequently relies on so-called PAC-reductions introduced by [Pitt and Warmuth, 1990] (cf. Chapter 7 by [Kearns and Vazzirani, 1994]). We will only consider a special type of PAC-reduction, which can be derived from the above introduced notion of a reduction:

Definition 12 *A reduction ρ is* efficient *if and only if $size(\rho(e))$ is bounded by $p(size(e))$ where p is a polynomial and ρ can be computed in polynomial time.*

When learning under $\in_x$ ***efficiently*** reduces to learning under $\in_y$, we will write that $\in_x \trianglelefteq \in_y$. It is clear that $\trianglelefteq$ is also transitive.

It is straightforward to prove that all reductions used in the proofs of theorems in Section 3, are efficient when natural and comparable size measures are used[8].

Thus the main result of this paper is

Theorem 6 *Efficient reductions*

1. *Learning from finite interpretations and extensional inductive logic programming efficiently reduce to learning from entailment;*

2. *Learning from entailment efficiently reduces to learning from finite partial interpretations, and vice versa,*

3. *Learning from entailment and from finite partial interpretations efficiently reduce to intensional inductive logic programming,*

4. *Intensional inductive logic programming efficiently reduces to learning from satisfiability.*

In graphical form Theorem 6 yields :

$$\begin{matrix} \in_i \\ \in_{ext,B} \end{matrix} \trianglelefteq \in_e = \in_{pi} \trianglelefteq \in_{int,B} \trianglelefteq \in_s$$

This result is important in the light of PAC-learning because :

Theorem 7 *If there is an efficient reduction from learning under $\in_x$ to learning under $\in_y$ and L_C is PAC-learnable under $\in_y$, then L_C is also PAC-learnable under $\in_x$.*

Proof: This follows from the observations that 1) if there is an efficient reduction from learning under $\in_x$ to learning under $\in_y$ then $(L_C, L_x, \in_x)$ PAC-reduces to $(L_C, L_y, \in_y)$; and 2) if $(L_C, L_x, \in_x)$ PAC-reduces to $(L_C, L_y, \in_y)$, and $(L_C, L_y, \in_y)$ is PAC-learnable then $(L_C, L_x, \in_x)$ is PAC-learnable.

1) follows directly from the definition of a PAC-reduction (see e.g. [Kearns and Vazzirani, 1994], p. 147), because

- an efficient reduction ρ can serve as an efficient instance transformation that maps examples in L_x on examples in L_y, and

- the existence of an image concept is trivial as the concept classes considered are identical,

[8]This implies that when learning from interpretations, the size of an interpretation takes into account (i.e. sums) the sizes of the set of true and the set of false facts, which contrasts with [De Raedt and Džeroski, 1994], who take into account only the size of the true facts. Using the sum is necessary because the reduction from $\in_i$ to e.g. $\in_e$ results in clauses that contain both true as well as false facts.

- the instance mapping preserves concept membership [9]

2) is a direct application of theorem 7.2. in [Kearns and Vazzirani, 1994], or the results by [Pitt and Warmuth, 1990]. $\square$

4.2 PAC-learning and logic

Computational learning theory has investigated the learnability of several classes of logical hypotheses, under various coverage notions.

First, k-CNF, the class of all CNF formulae that contain at most k literals per clause [Valiant, 1984], and jk-CT, the class of all CT formulae that contain at most k literals of size at most j per clause [De Raedt and Džeroski, 1994] are efficiently PAC-learnable from interpretations (both from positive and from positive and negative examples). We will now show that k-CNF is not efficiently PAC-learnable under entailment. Though one might consider learning k-CNF under entailment inappropriate from a PAC-learning point of view as membership testing under entailment is NP-hard, this theorem does show that learning k-CNF under entailment is not PAC-reducible to learning k-CNF under interpretations. This in turn suggests that learning under entailment is not only more general but also computationally harder than learning from interpretations.

Theorem 8 *k-CNF is not efficiently PAC-learnable under entailment for $k \geq 3$.*

Proof: Due to the results of [Pitt and Valiant, 1988] it suffices to show that finding a solution to the learning problem (the so-called consistency problem) is NP-hard.

The consistency problem can be used to solve the well-known NP-hard 3-SAT problem. 3-SAT is the problem of determining whether a 3-CNF formula is satisfiable or not.

Consider a 3-CNF formula $T = \wedge_{i=1}^{m}(l_{i,1} \vee l_{i,2} \vee l_{i,3})$ over n propositional predicates. Consider the equivalent learning problem, where $P = \{l_{i,1} \vee l_{i,2} \vee l_{i,3} \mid 1 \leq i \leq m\}$ and $N = \{\emptyset\}$ (where the only negative example is the unsatisfiable clause).

We still have to prove that the 3-CNF formula is satisfiable if and only if the learning problem has a solution: if the learning problem has a solution H, then $H \models T$ and $H \not\models \square$, therefore $T \not\models \square$; if T is satisfiable, then the learning problem has a solution, i.e. $H = T$. $\square$

Similarly, one can prove

Theorem 9 *jk-CT for $3 \leq k$ is not PAC-learnable under entailment*

Corollary 2 *k-CNF and jk-CT are not PAC-learnable under $\in_{int,B}$ and $\in_{s}$.*

Secondly, various classes of concepts have been investigated within the inductive logic programming paradigm, see [Cohen and Page, 1995; Kietz and Dzeroski, 1994; Cohen, 1995] for overviews. Most of these results concern definite clauses under $\in_{ext,B}$. All of the ***negative*** results for this settings carry over to learning under $\in_e$, $\in_{pi}$, $\in_{int,B}$ and $\in_s$ because of theorems 6 and 7. On the other hand, positive results for $\in_e$ or $\in_s$ carry over to $\in_i$.

[9] Though the usual definition of PAC-reduction requires that the instance mapping maps positives onto positives and negatives onto negatives (and hence that concept-membership is preserved, it is easily proven that result 2) also holds when positives are mapped onto negatives and vice versa (and hence that negated membership is preserved), cf. the proof of Theorem 7.2 by [Kearns and Vazzirani, 1994].

Finally, Horn-CNF, the class of CNF formulae that are Horn, are learnable using membership and equivalence queries from interpretations [Angluin ***et al.***, 1992] and from entailment [Frazier and Pitt, 1993]. This also means that they are PAC-learnable if membership questions are available[10]. At present, it remains an open question as to whether these results can be upgraded towards restricted sets of first order logic and whether they would carry over to learning under $\in_{int,B}$ or $\in_s$. On the other hand, due to the equivalence of $\in_e$ and $\in_{pi}$, Horn-CNF should be learnable in the same manner from partial interpretations.

5 Related work

The presented work is related to and motivated by some of the results by [Angluin, 1987] and [Frazier and Pitt, 1993] who study the relation between learning from entailment and learning from interpretations when membership and equivalence queries are available. However, our results do not assume that queries are available.

Secondly, our results are also related to those by [Wrobel and Džeroski, 1995] who study the relation among several inductive logic programming settings. In particular, they studied the influence of testing coverage at the local level (i.e. representing each example by a seperate logical theory) or at the global level (i.e. representing the example set as a whole by a single logical theory), and the differences between predictive and descriptive inductive logic programming. Wrobel and Džeroski also propose to use $\in_s$ to test for coverage (in one of their settings). However, they did not specify the language of examples and hypotheses, which is crucial for obtaining our results. From this point of view, the main novelty in learning from satisfiability is the use of full clausal theories to represent hypotheses and examples.

Finally, several of our results relate to Peter Flach's inquiry into the logic of induction [Flach, 1995; Flach, 1992; Flach, 1994]. Flach's work provides a normative semantic account of inductive reasoning in which meta-rules are used to describe various properties of inductive reasoning. Flach distinguishes explanatory from confirmatory induction using these meta-rules. Explanatory induction is related to learning from entailment, whereas confirmatory induction is closer to learning from interpretations or learning from satisfiability. The main difference between Flach's work and ours, is that Flach assumes that the example ***set*** is represented by a single logical theory, whereas in our framework each example corresponds to a logical formula. This is important as in Flach's work positive as well as negative examples are handled identically (though a positive example would be a true clause, and a negative one the negation of a clause). This not only complicates the logic[11] but also makes it hard to view Flach's setting as concept-learning, because the latter is typically concerned with positive and negative examples as well as with classification. This is further illustrated by Flach's notion of learning from satisfiability, which requires that $H \wedge E \not\models \Box$ where E is the complete example set. Flach views this as confirmatory induction, of which the prime characteristic is that it is not classification oriented. Our framework shows that it is feasible and interesting to adapt this notion for classification-oriented concept-learning.

[10]A membership question asks an oracle whether or not an example belongs to the target concept.

[11]To require that negative statements are entailed, Flach needs to rely on non-monotonic logic. Furthermore, negative examples are sometimes added to the background theory, which seems counter-intuitive.

6 Conclusions

Our results allow us to formulate answers to the open questions in the introduction. First, the relation among ***learning from interpretations*** and ***learning from entailment***, raised in various forms within computational learning theory and inductive logic programming [Frazier and Pitt, 1993; De Raedt and Džeroski, 1994; Muggleton and De Raedt, 1994; Wrobel and Džeroski, 1995; De Raedt and Lavrač, 1996] is now clarified. Secondly, whereas attribute value learning techniques have mostly learned from interpretations and inductive logic programming from entailment, our results indicate that even in the propositional case inductive logic programming is more general and harder (as illustrated by k-CNF). Thirdly, the result on k-CNF indicates that the normal inductive logic programming setting (as formalized in learning from entailment) is computationally harder than the non-monotonic setting (as formalized by [De Raedt and Džeroski, 1994]). This last contribution confirms some of the earlier intuitions about the differences between the non-monotonic setting and normal inductive logic programming as formulated by e.g. [Muggleton and De Raedt, 1994; De Raedt and Lavrač, 1996] and between weak and strong induction [Flach, 1992].

Finally, we also leave open a number of questions. First, about learning under satisfiability, one may wonder whether there exist still reasonable classes that are learnable and also, what algorithms can be used to do so. Some of the latter issues are addressed by [De Raedt and Dehaspe, 1996]. Second, our main results concern full clausal logic, whereas in practice one mostly considers Horn logic only. As a consequence, if one requires that examples are Horn-clauses when learning from entailment, then the relation to learning from interpretations is less clear. Third, within inductive logic programming, most of the learnability results (see [Cohen and Page, 1995]) are very specific within our framework as they concern $\in_{\boldsymbol{ext,B}}$. Furthermore, though they typically assume a bound j on the arity of predicates in the background theory, they do not impose a bound on the arity of the predicates to be learned (because otherwise the learning task is considered trivial). An alternative would be to simply employ learning from entailment with one size measure on the length of clauses (and the possibility of also using bounds on the arity of predicates to be learned). This type of learning has been addressed by the algorithmic learning theory community (cf. [Ishizaka ***et al.***, 1994; Miyano ***et al.***, 1991; Mukouchi and Arikawa, 1993]).

Acknowledgements

The author is supported by the Belgian National Fund for Scientific Research and the ESPRIT IV project no. 20237 Inductive Logic Programming II. He is especially grateful to Peter Flach for inspiring discussions, to Marc Denecker and Sašo Džeroski for technical help, and to Maurice Bruynooghe, Hendrik Blockeel, Luc Dehaspe, David Page, and Wim Van Laer for commenting on earliers versions of this note.

References

[Angluin ***et al.***, 1992] D. Angluin, M. Frazier, and L. Pitt. Learning conjunctions of horn clauses. *Machine Learning*, 9:147–162, 1992.

[Angluin, 1987] D. Angluin. Learning propositional horn sentences with hints. Technical Report YALEU/DCS/RR-590, Yale University, 1987.

[Cohen and Page, 1995] W.W. Cohen and D. Page. Polynomial learnability and inductive logic programming: Methods and results. *New Generation Computing*, 13, 1995.

[Cohen, 1995] W.W. Cohen. PAC-learning non-recursive Prolog clauses. *Artificial Intelligence*, 79:1–38, 1995.

[De Raedt and Dehaspe, 1996] L. De Raedt and L. Dehaspe. Learning from satisfiability. Technical Report KUL-CW, Department of Computer Science, Katholieke Universiteit Leuven, 1996. forthcoming.

[De Raedt and Dehaspe, 1997] L. De Raedt and L. Dehaspe. Clausal discovery. *Machine Learning*, 26:99–146, 1997.

[De Raedt and Džeroski, 1994] L. De Raedt and S. Džeroski. First order jk-clausal theories are PAC-learnable. *Artificial Intelligence*, 70:375–392, 1994.

[De Raedt and Lavrač, 1996] L. De Raedt and N. Lavrač. Multiple predicate learning in two inductive logic programming settings. *Journal of the Interest Group in Pure and Applied Logics*, 4(2):227–254, 1996.

[De Raedt and Van Laer, 1995] L. De Raedt and W. Van Laer. Inductive constraint logic. In *Proceedings of the 5th Workshop on Algorithmic Learning Theory*, volume 997 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1995.

[De Raedt, 1996] L. De Raedt. Induction in logic. In R.S. Michalski and Wnek J., editors, *Proceedings of the 3rd International Workshop on Multistrategy Learning*, pages 29–38, 1996.

[Fensel *et al.*, 1995] D. Fensel, M. Zickwolff, and M. Wiese. Are substitutions the better examples? In L. De Raedt, editor, *Proceedings of the 5th International Workshop on Inductive Logic Programming*, 1995.

[Flach, 1992] P. Flach. A framework for inductive logic programming. In S. Muggleton, editor, *Inductive logic programming*. Academic Press, 1992.

[Flach, 1994] P.A. Flach. Inductive logic programming and philosophy of science. In S. Wrobel, editor, *Proceedings of the 4th International Workshop on Inductive Logic Programming*, volume 237 of *GMD-Studien*, Sankt Augustin, Germany, 1994. Gesellschaft für Mathematik und Datenverarbeitung MBH.

[Flach, 1995] P. Flach. *An inquiry concerning the logic of induction*. PhD thesis, Tilburg University, Institute for Language Technology and Artificial Intelligence, 1995.

[Frazier and Pitt, 1993] M. Frazier and L. Pitt. Learning from entailment. In *Proceedings of the 9th International Conference on Machine Learning*, 1993.

[Genesereth and Nilsson, 1987] M. Genesereth and N. Nilsson. *Logical foundations of artificial intelligence.* Morgan Kaufmann, 1987.

[Greiner *et al.*, 1996] R. Greiner, A.J. Grove, and A. Kogan. Exploiting the ommission of irrelevant data. In L. Saitta, editor, *Proceedings of the 13th International Conference on Machine Learning*, pages 207–215. Morgan Kaufmann, 1996.

[Ishizaka *et al.*, 1994] H. Ishizaka, H. Arimura, and T. Shinohara. Finding tree patterns consistent with positive and negative examples with queries. In S. Arikawa and K.P. Jantke, editors, *Proceedings of the Joint International Workshop on Analogical and Inductive Inference and Algorithmic Learning Theory*, Lecture Notes in Artificial Intelligence, pages 317–332. Springer-Verlag, 1994.

[Kearns and Vazzirani, 1994] M. Kearns and U. Vazzirani. *An introduction to Computational learning theory.* The MIT Press, 1994.

[Kietz and Dzeroski, 1994] J.U. Kietz and S. Dzeroski. Inductive logic programming and learnability. *Sigart Bulletin*, 5(1):22–32, 1994.

[Lloyd, 1987] J.W. Lloyd. *Foundations of logic programming.* Springer-Verlag, 2nd edition, 1987.

[Mitchell, 1982] T.M. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.

[Miyano *et al.*, 1991] S. Miyano, A. Shinohara, and T. Shinohara. Which classes of elementary formal systems are polynomial-time learnable ? In S. Arikawa, A. Maruoka, and T. Sato, editors, *Proceedings of the 2nd Workshop on Algorithmic Learning Theory*, 1991.

[Muggleton and De Raedt, 1994] S. Muggleton and L. De Raedt. Inductive logic programming : Theory and methods. *Journal of Logic Programming*, 19,20:629–679, 1994.

[Muggleton and Feng, 1990] S. Muggleton and C. Feng. Efficient induction of logic programs. In *Proceedings of the 1st conference on algorithmic learning theory*, pages 368–381. Ohmsma, Tokyo, Japan, 1990.

[Mukouchi and Arikawa, 1993] Y. Mukouchi and S. Arikawa. Inductive inference machines that can refute hypothesis spaces. In K.P. Jantke, S. Kobayashi, E. Tomita, and T. Yokomori, editors, *Proceedings of the 3rd Workshop on Algorithmic Learning Theory*, volume 744 of *Lecture Notes in Artificial Intelligence*, pages 123–136. Springer-Verlag, 1993.

[Pitt and Valiant, 1988] L. Pitt and L. Valiant. Computational limitations on learning from examples. *Journal of the Association for Computing Machinery*, 35:965–984, 1988.

[Pitt and Warmuth, 1990] L. Pitt and M. Warmuth. Prediction-preserving reducibility. *Journal of Computer and System Science*, 41:430–467, 1990.

[Quinlan, 1990] J.R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

[Valiant, 1984] L. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.

[Wrobel and Džeroski, 1995] S. Wrobel and S. Džeroski. The ILP description learning problem: Towards a general model-level definition of data mining in ILP. In *Proceedings of the Fachgruppentreffen Maschinelles Lernen*, 1995.