Machine Learning 35, 207–224 (1999)
© 1999 Kluwer Academic Publishers. Manufactured in The Netherlands.

A Dichotomy Theorem for Learning Quantified Boolean Formulas

VÍCTOR DALMAU dalmau@lsi.upc.es
Departament LSI, Universitat Politècnica de Catalunya, Campus Nord, Mòdul C5, Jordi Girona Salgado, 1-3, Barcelona 08034, Spain

Editor: John Shawe-Taylor

Abstract. We consider the following classes of quantified boolean formulas. Fix a finite set of basic boolean functions. Take conjunctions of these basic functions applied to variables and constants in arbitrary ways. Finally quantify existentially or universally some of the variables. We prove the following *dichotomy theorem*: For any set of basic boolean functions, the resulting set of formulas is either polynomially learnable from equivalence queries alone or else it is not PAC-predictable even with membership queries under cryptographic assumptions. Furthermore, we identify precisely which sets of basic functions are in which of the two cases.

Keywords: computational learning theory, quantified boolean formulas, dichotomy theorems

1. Introduction

The problem of learning an unknown boolean formula under some determined protocol has been widely studied. It is well known that, even restricted to propositional formulas, the problem is hard (Angluin & Kharitonov, 1995; Kearns & Valiant, 1994) in the usual learning models. Therefore, researchers have attempted to learn subclasses of propositional boolean formulas, specially inside CNF and DNF. For example, k-DNF formulas, k-term DNF formulas, monotone-DNF formulas, Horn formulas, and their dual counterparts (Angluin, 1988; Berggren, 1993; Angluin, Frazier, & Pitt, 1992) have all been shown exactly learnable using membership and equivalence queries in Angluin's model (Angluin, 1988) while the question of whether DNF formulas are learnable is still open. The more powerful formalism of predicate logics is used in several applications of learning in artificial intelligence and knowledge representation but its study, from the computational learning theory point of view, is recent. See Maass & Turán (1995) and the further references in that paper.

In this paper, we study the complexity of learning some subclasses of quantified boolean formulas called quantified boolean formulas over a basis S. These formulas are still propositional formulas but augmented with the additional capability of quantification.

Let $S = \{R_1, \ldots, R_m\}$ be a finite set of logical relations. Define an $\exists\forall$-Formula(S) to be any boolean formula formed by quantified conjunctions of any number of clauses of the form $R_i(\xi_1, \ldots, \xi_k)$, where $\xi_1, \ldots, \xi_k$ are variables or constants and k is the rank of R_i.

Example 1. Consider the problem of learning a boolean formula formed by a quantified conjunction of clauses with three literals per clause. Every such formula can be expressed as a formula in $\exists\forall$-Formula(S) with the set of logical relations $S = \{R_0, R_1, R_2, R_3\}$, defined by:

$$\begin{aligned} R_0(x, y, z) &\equiv x \vee y \vee z, \\ R_1(x, y, z) &\equiv \bar{x} \vee y \vee z, \\ R_2(x, y, z) &\equiv \bar{x} \vee \bar{y} \vee z, \\ R_3(x, y, z) &\equiv \bar{x} \vee \bar{y} \vee \bar{z}. \end{aligned}$$

The main result of this paper characterizes the complexity of learning $\exists\forall$-Formula(S) for every finite set S of logical relations. The most striking feature of this characterization is that for any S, $\exists\forall$-Formula(S) is either polynomially learnable with equivalence queries alone or, under some cryptographic assumptions, not polynomially predictable even with membership queries. In fact, for the hardness result, it is enough to consider formulas with existential quantifiers only or without constants. This dichotomy is somewhat surprising since one might expect that any such large and diverse family of concept classes would include some representatives of the many intermediate learning models such as exact learning with equivalence and membership queries, PAC learning with and without membership queries and PAC-prediction without membership queries.

Furthermore, we give an interesting classification of the polynomially learnable classes. We show that, in a sense that will be precised later, $\exists\forall$-Formula(S) is polynomially learnable if and only if at least one of the following conditions holds:

(a) Every relation in S is definable by a CNF formula in which each clause has at most 2 literals.
(b) Every relation in S is definable by a CNF formula in which each clause has no negated variables or has at most one negated variable and at most one affirmed variable.
(c) Every relation in S is definable by a CNF formula in which each clause has no affirmed variables or has at most one affirmed variable and at most one negated variable.
(d) Every relation in S is the set of solutions of a system of linear equations over the two-element field $\{0, 1\}$.

It is interesting to compare this classification with some previous known results about the learnability of quantifier-free formulas. First, notice that whereas k-CNF are polynomially learnable with equivalence queries, the equivalent result for quantified formulas is only valid for $k \leq 2$. In fact, the gap beetween the learnability of quantifier-free and quantified formulas is even wider: whereas for every finite set S of logical relations, the class of quantifier-free formulas in $\exists\forall$-Formula(S) is polynomially learnable with equivalence queries, the full class of formulas contains only few learnable subclasses.

It is also interesting to point out that whereas some classes of quantifier-free formulas over basis of infinite size are learnable, they turn out to be non-learnable in general if we allow quantification even restricting the basis to some finite subset. Horn formulas are an example of such feature. Notice that, whereas membership queries are of no help in the learnability of quantified boolean formulas over a finite basis they turn out to be needed to learn some classes of formulas obtained from infinite basis, such as monotone CNF or

Horn CNF. This fact seems to suggest that membership queries make a difference exactly when we are dealing with formulas built from basic relations of arbitrary arity.

A few but not many Dichotomy results in complexity theory are already known. The first one is a dichotomy result for the generalized satisfiability decision problem by Schaefer (1978). The others known to us concern the H-coloring of graphs (Hell & Nešetřil, 1990), the subgraph homeomorphism (Fortune, Hopcroft, & Wyllie, 1980), the decomposition of graphs (Dor & Tarsi, 1997), the inverse generalized satisfiability problem (Kavvadias & Sidderi, 1996), the generalized satisfiability counting problem (Creignou & Hermann, 1996), and the approximability of minimization and maximization problems (Creignou, 1995; Khanna, Sudan, & Trevisan, 1997; Khanna, Sudan, & Williamson, 1997). It is remarkable that most of the dichotomy results shown before are in the framework of generalized satisfiability problems proposed by Schaefer. Our result is inspired as well by the framework and techniques of Schaefer which allow us to compare the complexity of different problems on generalized quantified boolean formulas. For example, from Schaefer's Dichotomy Theorem (Schaefer, 1978) and the Dichotomy Theorem of this paper can be inferred that, in this framework, learnability is slightly harder than satisfiability.

The aim of this paper is to study the complexity of learning generalized quantified boolean formulas, but some intermediate results are interesting in themselves. In particular, the technique used in the semantic characterization of weakly antimonotone logical relations could be useful in characterizing other logical relations defined as the conjunction of some restricted kinds of clauses.

2. Definitions and notation

2.1. Learning models

Most of the terminology used in this section comes from Angluin & Kharitonov (1995). Let X denote $\{0, 1\}^*$; binary strings will represent both examples and concept names. Let x be a string, $|x|$ denotes its length, and for every constant $b \in \{0, 1\}$, $|x|_b$ denotes the number of occurrences of b in x. For any natural number n, $X^{[n]} = \{x \in X : |x| \leq n\}$. A *representation of concepts* (or *representation class*) $\mathcal{C}$ is any subset of $X \times X$. We interpret an element $\langle u, x\rangle$ of $X \times X$ as consisting of a *concept name* u and an *example* x. The example x is a member of the concept u if and only if $\langle u, x\rangle \in \mathcal{C}$. Define the *concept represented by* u as

$$K_{\mathcal{C}}(u) = \{x : \langle u, x\rangle \in \mathcal{C}\}$$

The *set of concepts represented by* $\mathcal{C}$ is

$$\{K_{\mathcal{C}}(u) : u \in X\}$$

In this paper we use two models of learning, both of which are fairly standard: Angluin's model of exact learning from equivalence queries (Angluin, 1988) and the model of PAC-prediction with membership queries as defined by Angluin & Kharitonov (1995).

Let $\mathcal{H}$ be a representation class. A learning algorithm with queries is an algorithm A that takes as input a bound s on the size of the target concept representation and a bound n on the length of the examples. It may make any number of queries or requests, the responses to which are determined by the unknown target concept c. A must eventually halt with an output concept name v. The concept $K_{\mathcal{H}}(v)$ is interpreted as A's guess of the target concept. The most common kinds of queries are the *membership* and the *equivalence* queries. A membership query takes a string $x \in X$ as input and returns 1 if $x \in c$ and 0 otherwise. An equivalence query takes a concept name h as input and returns *yes* if $c = K_{\mathcal{H}}(h)$ and a counterexample $x \in c \bigtriangleup K_{\mathcal{H}}(h)$ otherwise. *A runs in polynomial time* if its running time (counting one step for oracle call) is bounded by a polynomial in s and n.

We say that A successfully *exactly learns* a representation of concepts $\mathcal{C}$, if and only if for all positive integers s, n, for all concept names $u \in X^{[s]}$, when A is run with inputs s and n, and oracles determined by $c = K_{\mathcal{C}}(u)$, A outputs a concept name v such that $c = K_{\mathcal{H}}(v)$. If $\mathcal{C} = \mathcal{H}$ we say that A learns *properly* $\mathcal{C}$, otherwise we say that A learns *improperly* $\mathcal{C}$. A representation of concepts $\mathcal{C}$ is *polynomially learnable* if and only if there is a learning with queries algorithm A that runs in polynomial time and successfully learns exactly $\mathcal{C}$.

A *prediction with membership algorithm*, or *pwm-algorithm*, is a possibly randomized algorithm A that takes as input a bound s on the size of the target concept representation, a bound n on the length of examples, and an accuracy bound ϵ. It may make three different kinds of oracle calls, the responses to which are determined by the unknown target concept c and the unknown distribution D on $X^{[n]}$, as follows:

- A membership query takes a string $x \in X$ as input and returns 1 if $x \in c$ and 0 otherwise.
- A request for a random classified example takes no input and returns a pair $\langle x, b \rangle$, where x is a string chosen independently according to D and $b = 1$ if $x \in c$ and $b = 0$ otherwise.
- A request for an element to predict takes no input and returns a string x chosen independently according to D.

A may make any number of membership queries or requests for random classified examples. However, A must eventually make one and only one request for an element to predict and eventually halt with an output of 1 or 0 without making any further oracle calls. The output is interpreted as A's guess of how the target concept classifies the element returned by the request for an element to predict. *A runs in polynomial time* if its running time (counting one step per oracle call) is bounded by a polynomial in s, n, and $1/\epsilon$.

We say that A successfully *predicts* a representation of concepts $\mathcal{C}$ if and only if for all positive integers s and n, for all positive rationals ϵ, for all concept names $u \in X^{[s]}$, when A is run with inputs s, n, and ϵ, and oracles determined by $c = K_{\mathcal{C}}(u)$ and D, A asks membership queries that are in X and the probability is at most ϵ that the output of A is not equal to the correct classification of x by $K_{\mathcal{C}}(u)$, where x is the string returned by the (unique) request for an element to predict. We can say that A predicts $\mathcal{C}$ in PAC sense, with the additional help of membership queries. See Valiant (1984) for a formal definition of the PAC model.

A representation of concepts $\mathcal{C}$ is *polynomially predictable with membership queries* if and only if there is a *pwm*-algorithm A that runs in polynomial time and successfully predicts $\mathcal{C}$. If a representation of concepts is learnable in polynomial time with membership and equivalence queries then it is polynomially predictable with membership queries.

To compare the difficulty of learning problems in the prediction model we use the *prediction-preserving reducibility with membership queries* as defined by Angluin & Kharitonov (1995). It is denoted by $\leq_{pwm}$ and it extends Pitt & Warmuth's (1990) *prediction-preserving reducibility* to the presence of membership queries.

Definition 2. Let $\mathcal{C}$ and $\mathcal{C}'$ be representations of concepts. Let $\perp$ and $\top$ be elements not in X. Then $\mathcal{C}$ is *pwm-reducible* to $\mathcal{C}'$, denoted $\mathcal{C} \leq_{pwm} \mathcal{C}'$, if and only if there exist three mappings g, f, and h with the following properties:

1. There is a nondecreasing polynomial q such that for all natural numbers s and n and for $u \in X^{[s]}$, $g(s, n, u)$ is a string u' of length at most $q(s, n, |u|)$.
2. For all natural numbers s and n, for every string $u \in X^{[s]}$, and for every $x \in X^{[n]}$, $f(s, n, x)$ is a string x' and $x \in K_{\mathcal{C}}(u)$ if and only if $x' \in K_{\mathcal{C}'}(g(s, n, u))$. Moreover, f is computable in time bounded by a polynomial in s, n, and $|x|$, hence there exists a nondecreasing polynomial t such that $|x'| \leq t(s, n, |x|)$.
3. For all natural numbers s and n, for every string $u \in X^{[s]}$, and for every $x' \in X$, $h(s, n, x')$ is either $\perp$, $\top$, or a string $x \in X$. If $h(s, n, x') = \top$ then $x' \in K_{\mathcal{C}'}(g(s, n, u))$, if $h(s, n, x') = \perp$ then $x' \notin K_{\mathcal{C}'}(g(s, n, u))$, and otherwise $x' \in K_{\mathcal{C}'}(g(s, n, u))$ if and only if $x \in K_{\mathcal{C}}(u)$. Moreover, h is computable in time bounded by a polynomial in s, n, and $|x'|$.

In (2), and independently in (3), the expression "$x \in K_{\mathcal{C}}(u)$" can be replaced with "$x \notin K_{\mathcal{C}}(u)$", as discussed in Angluin & Kharitonov (1995).

The only properties of this reducibility that are needed in this paper were shown in Angluin & Kharitonov (1995):

Lemma 3. *The pwm-reduction is transitive, i.e., let $\mathcal{C}, \mathcal{C}'$ and $\mathcal{C}''$ be representations of concepts, if $\mathcal{C} \leq_{pwm} \mathcal{C}' \leq_{pwm} \mathcal{C}''$ then $\mathcal{C} \leq_{pwm} \mathcal{C}''$.*

Lemma 4. *Let $\mathcal{C}$ and $\mathcal{C}'$ be representations of concepts. If $\mathcal{C} \leq_{pwm} \mathcal{C}'$ and $\mathcal{C}'$ is polynomially predictable with membership queries, then $\mathcal{C}$ is also polynomially predictable with membership queries.*

2.2. *Logical preliminaries*

Let $V = \{x_1, x_2, \ldots\}$ be an infinite set of boolean variables. A literal is a variable or its negation. A clause is a disjunction of literals. An assignment is a vector in X. For any assignment $t \in X$ and for any integer j, $t[j] \in \{0, 1\}$ denotes the jth component of t. Logical operators ($\vee$, $\wedge$, $\neg$) can also be applied to assignments meaning that they are operated component-wise. Given two assignments t_1 and t_2, $t_1 t_2$ denotes the assignment obtained concatenating t_1 and t_2.

A logic relation of rank k (k integer) is a subset of $\{0, 1\}^k$. There exists an unique assignment of length 0, we call it λ.

We use the term *formula* in a large sense, to mean any well-formed formula, formed from variables, constants, logical connectives, parentheses, logical relation symbols, and existential and universal quantifiers.

Let $S = \{R_1, \ldots, R_m\}$ be any finite set where each R_i is a logical relation of rank k_i. R_i denotes both the logical relation and its symbol. The set of formulas formed by conjunctions of relations in S with constants is denoted Formula(S). Specifically, Formula(S) is the smallest set of formulas such that:

- For all $R \in S$ of rank k, $R(y_1, \ldots, y_k) \in$ Formula(S) where $y_i \in V \cup \{0, 1\}$ for $1 \leq i \leq k$.
- For all $F, G \in$ Formula(S), $F \wedge G \in$ Formula(S).

The set of *quantified boolean formulas over the basis S*, denoted by $\exists\forall$-Formula(S), is the smallest set of formulas such that:

- For all $F \in$ Formula(S), $F \in \exists\forall$-Formula(S).
- For all $F \in \exists\forall$-Formula(S) and for all $\xi \in V$, $\exists\xi F$ and $\forall\xi F$ are in $\exists\forall$-Formula(S).

We call $\exists$-Formula(S) the subset of $\exists\forall$-Formula(S) that we obtain if we allow only existential quantifiers.

Each formula F defines a logical relation $[F]$ if we apply the usual semantics of first-order logic and the variables are taken in lexicographical order. For every set of logical relations S we define Relation$(S) = \{[F] : F \in$ Formula$(S)\}$. Analogously, we define $\exists\forall$-Relation(S) and $\exists$-Relation(S) as the set of logical relations obtained from formulas in $\exists\forall$-Formula(S) and $\exists$-Formula(S) respectively.

For any set of formulas $\mathcal{F}$, we define $\mathcal{C}_{\mathcal{F}}$ as the representation of concepts formed from formulas in $\mathcal{F}$. More precisely, $\mathcal{C}_{\mathcal{F}}$ contains all the tuples of the form $\langle f, x\rangle$ where f represents a formula in $\mathcal{F}$ and x is a model satisfying f.

Example 5. Consider the basis introduced in Example 1. Let F be following formula:

$$\begin{aligned} F = \forall x_1 \exists x_2 \forall x_3 \quad & R_1(x_1, x_2, x_3) \\ & \wedge R_1(x_4, x_3, x_2) \\ & \wedge R_2(x_4, x_5, 0) \\ & \wedge R_3(1, x_6, x_2) \end{aligned}$$

F is a formula in $\exists\forall$-Formula(S) over the free variables x_4, x_5, and x_6. $[F]$ contains exactly all the assignments over these variables satisfying F.

$$\begin{aligned} [F] = \{ \quad & \langle 0, 0, 0\rangle \\ & \langle 0, 1, 0\rangle \\ & \langle 1, 0, 0\rangle \quad \} \end{aligned}$$

F can also be regarded as a concept in $\mathcal{C}_{\exists\forall-\text{Formula}(S)}$ $[F]$, being its set of examples.

A clause is *bijunctive* if it has at most 2 literals. A clause is *horn* (resp. *antihorn*) if it has at most one affirmed (resp. negated) variable. A clause is *weakly monotone* (resp. *weakly antimonotone*) if it is (i) the disjunction of affirmed variables (resp. negated variables) or (ii) the disjunction of at most two literals with at most one negated (resp. affirmed) variable. That is, a weakly antimonotone clause is a horn clause where we only allow rules like $(y_1 \ldots y_n \rightarrow 0)$ and $(y_i \rightarrow y_j)$ where every y_i is a variable or a constant. The logical relation R of rank k is bijunctive (resp. horn, antihorn, weakly monotone, weakly antimonotone) if $R(x_1, \ldots, x_k)$ is logically equivalent to some CNF formula where each clause is bijunctive (resp. horn, antihorn, weakly monotone, weakly antimonotone). The logical relation R of rank k is *affine* if $R(x_1, \ldots, x_k)$ is logically equivalent to some system of linear equations over the two-element field $\{0, 1\}$.

We can extend the definitions above to formulas and sets of relations: The formula F is bijunctive (resp. horn, antihorn, weakly monotone, weakly antimonotone, affine) if $[F]$ is bijunctive (resp. horn, antihorn, weakly monotone, weakly antimonotone, affine). The set S of logical relations is bijunctive (resp. horn, antihorn, weakly monotone, weakly antimonotone, affine) if every $R \in S$ is bijunctive (resp. horn, antihorn, weakly monotone, weakly antimonotone, affine).

The *degree* of a logical relation R of rank k, is the minimum value $d \leq k$ such that R can be expressed as a d-CNF formula. Analogously, the degree of a formula F is the degree of the logical relation $[F]$. The *degree* of a finite set of logical relations S is the maximum of the degrees of all relations in S.

3. The dichotomy theorem

This section states and proves the main result of this paper:

Theorem 6 (Dichotomy Theorem for Learnability). *Let S be a finite set of logical relations. If S satisfies one of the conditions* (a)–(d) *below, then* $\mathcal{C}_{\exists\forall\text{-}Formula(S)}$ *is polynomially exactly learnable with improper equivalence queries. Otherwise,* $\mathcal{C}_{\exists\text{-}Formula(S)}$ *is not polynomially predictable with membership queries under the assumption that public key encryption systems secure against chosen ciphertext attack exist.*

(a) *S is bijunctive.*
(b) *S is affine.*
(c) *S is weakly monotone.*
(d) *S is weakly antimonotone.*

We refer the reader to Angluin & Kharitonov (1995) for definitions of the cryptographic concepts.

Schaefer (1978) proves a similar dichotomy theorem for the satisfiability of ∃∀-Formula (S). He shows that this problem is polynomial-time solvable if and only if S is bijunctive, horn, antihorn or affine. Otherwise the problem is PSPACE-complete (NP-complete if we take only ∃-Formula(S)). We can note here that if a basis S does not fall in the classes in

Schaefer's theorem then the representation of concepts $\exists\forall$-Formula(S) is not polynomial-time evaluable, i.e., given an example and a representation of a concept, it is not possible to decide in polynomial time whether the example is a member of the concept, unless P = PSPACE.

From the comparison of both theorems it follows that, in this framework, learning is slightly harder than deciding satisfiability. More precisely, satisfiability when S is horn is polynomial-time decidable, but for polynomial time learnability we must guarantee that S falls in a more restricted class, namely the weakly antimonotone sets.

Observe that for the negative results only existential quantifiers are needed. In Dalmau (1997) a proof of the negative result, in the case that constants are not allowed, can be found. Therefore, the main theorem does not change at all if we are restricted to formulas without universal quantifiers or without constants.

From now on, a set of logical relations S will be called a *basic set* if S is bijunctive, weakly monotone, weakly antimonotone, or affine and the representation of concepts $\exists\forall$-Formula(S) formed from a basic set S will be called *basic representation of concepts*.

Here is a bird's eye view of the proof.

(a) The efficient learnability of basic classes follows from the following results
   (i) the expressive power of formulas over basic sets is essentially the same with and without using quantifiers, and
   (ii) in consequence the learning problem can be reduced to quantifier-free formulas that are contained in some already known learnable classes.

(b) For the non-learnability results, we show that
   (i) the quantified formulas over any non-basic set can express a double implication $xy \to z$ or its dual.
   (ii) Then we show that this implication is enough to simulate boolean circuits, which are hard to learn under cryptographic assumptions.

The rest of the paper is structured as follows:

Section 3.1 contains exclusively results about relations and their expressivity power. No learning notions are involved. This section is organizated as follows: Section 3.1.1 contains item (a(i)). Section 3.1.2 contains some characterizations of some classes in semantic terms, that is, in terms of what elements are in the relation, rather that in terms of defining formulas as in the definitions. These alternative semantic definitions are easier to handle in Section 3.1.3 wich contains item (b(i)). Finally, item (a(ii)) is shown in Section 3.2 and Section 3.3 contains item (b(ii)).

3.1. Results in logic

3.1.1. Closure under quantification. In this section, we show that some sets of logic relations are "closed" under quantification. More precisely, we prove that if a set S of logical relations is bijunctive, weakly monotone, weakly montonone of affine, then quantifiers do not help to obtain a more reduced representation. Later on, we will use this property to show the learnability of quantified boolean formulas constructed using these families of basis.

Let us state the main result of this section:

Lemma 7. *If S is a bijunctive* (*resp. weakly monotone of degree d, weakly antimonotone of degree d, affine*) *set of logical relations then $\exists\forall$-Formula(S) is bijunctive* (*resp. weakly monotone of degree d, weakly antimonotone of degree d, affine*).

The proof of this result is rather simple. The underlying idea consists in that, by simple substitutions, it is possible to eliminate the quantifiers from inside to outside without increasing too much the size of the formula. The same analysis can handle bijunctive, weakly monotone and weakly antimonotone relations, since they all have been defined as a CNF using a particular set of clauses. Affine basis require some particular treatment instead.

We begin with some claims of easy proof:

Let F be a formula over V, let $\xi \in V$ be a variable and w a literal or a constant. We define $F[{\xi \atop w}]$ as the formula formed from F by replacing each occurrence of ξ by w. If $W \subset V$ is a set of variables then $F[{W \atop w}]$ denotes the result of substituting w for every occurrence of every variable in W. Multiple substitutions are denoted by expressions such as $F[{V \atop w}, {V' \atop w'}, {V'' \atop w''}]$ with obvious meaning.

Claim 8. *Let $x \in V$ be a variable. If F is a bijunctive* (*resp. weakly monotone of degree d, weakly antimonotone of degree d, affine*) *formula then $F[{x \atop 0}]$ and $F[{x \atop 1}]$ are bijunctive* (*resp. weakly monotone of degree d, weakly antimonotone of degree d, affine*).

Let $C = x \vee C'$ and $D = \bar{x} \vee D'$ be two clauses, where C' is the rest of clause C, and similarly for D and D'. That is, the two clauses contain two opposite literals. Then the clause $C' \vee D'$, containing all literals of the two clauses except for the two opposite literals is called the *resolvent* of C and D with respect to the variable x, denoted by $\mathcal{R}(C, D, x)$.

Claim 9. *Let $x \in V$ be a variable. Let C_x and $C_{\bar{x}}$ be a pair of clauses that contain the literal x and $\bar{x}$ respectively. If C_x and $C_{\bar{x}}$ are bijunctive* (*resp. weakly monotone of degree d, weakly antimonotone of degree d*), *then $\mathcal{R}(C_x, C_{\bar{x}}, x)$ is bijunctive* (*resp. weakly monotone of degree d, weakly antimonotone of degree d*).

We note here that in the case of general horn (resp. antihorn) clauses, although the resolvent of two horn (resp. antihorn) clauses is a horn (resp. antihorn) clause, we cannot guarantee that degree does not increase.

Claim 10. *For every variable $x \in V$ and for every formula F, $\forall x F \equiv F[{x \atop 0}] \wedge F[{x \atop 1}]$.*

Claim 11. *For every variable $x \in V$ and for every CNF formula F such that $F \equiv \bigwedge_{C \in G_x \cup G_{\bar{x}} \cup G} C$ where G_x, $G_{\bar{x}}$, and G are sets of clauses that contain the literal x, the literal $\bar{x}$, and none of them respectively, the following equivalence holds: $\exists x F \equiv \bigwedge_{C_x \in G_x, C_{\bar{x}} \in G_{\bar{x}}} \mathcal{R}(C_x, C_{\bar{x}}, x) \wedge \bigwedge_{C \in G} C$.*

From Claims 8–11 we can derive the following lemma:

Lemma 12. *Let F be a bijunctive (resp. weakly monotone of degree d, weakly antimonotone of degree d) formula. Then, for every variable $x \in V$, $\forall x F$ and $\exists x F$ are both bijunctive (resp. weakly monotone of degree d, weakly antimonotone of degree d) formulas.*

The analogous result for affine functions is proved differently.

Lemma 13. *Let F be an affine formula. Then, for every variable $x \in V$, $\forall x F$ and $\exists x F$ are both affine.*

Proof: Let F be equivalent to a system of linear equations over the two element field $\{0, 1\}$. To prove the case $\forall x F$ we only need to apply Claims 8 and 10 or, even easier, note that $\forall x F \equiv 0$ if x appears in F, and $\forall x F \equiv F$ otherwise. So we only need to consider the case $\exists x F$. We can take some equation that contains the variable x, separate x in one side of the equation and substitute the other side in the rest of equations. This process gives us a system of equations equivalent to $\exists x F$. □

We can eliminate systematically the quantifiers from inside to outside preserving the bijunctivity (resp. weakly monotonicity of degree d, weakly antimonotonicity of degree d, affinity) obtaining Lemma 7.

3.1.2. Semantic characterizations. In this section, we give semantic characterizations of some sets of logical relations. Other semantic characterizations can be found in Schaefer (1978). Semantic characterizations are more convenient to our purposes than the definitions given in Section 2.2. This is because, as we will see in the next section, we are interested in the logical relations generated by some relation R that does *not* belong to a particular class. In these cases, using the semantic characterizations we can infer that there exist some assignment in R not satisfying determined property and establish consequences from this fact.

The following characterization of horn relations is well known (see Papadimitriou (1994), for example).

Lemma 14 (Papadimitriou, 1994. Problem 4.4.7.). *Let R be a logical relation. R is horn (resp. antihorn) if and only if for all $t, t' \in R$, $t \wedge t' \in R$ (resp. $t \vee t' \in R$).*

At this point we show a characterization of the weakly antimonotone relations. We need the following notation:

Let $t \in \{0, 1\}^k$ be an assignment of length k and let $T = \{i_1, \ldots, i_j\}$ be a set of j indices $1 \leq i_1 < \cdots < i_j \leq k$. The projection $t \mid T$ is defined to be the assignment of arity j given by $t \mid T = \langle t[i_1], \ldots, t[i_j]\rangle$. Analogously, for every relation R of rank k, the projection $R \mid T$ is defined to be the relation of arity j

$$R \mid T = \{t \mid T : t \in R\}$$

That is, $R \mid T = [\exists x_{l_1}, \ldots, \exists x_{l_{k-j}} R(x_1, \ldots, x_k)]$ where $\{l_1, \ldots, l_{k-j}\} = \{1, \ldots, k\} - T$, and therefore $R \mid T \in \exists$-Relation($\{R\}$).

We have $t \mid \emptyset = \lambda$ and $R \mid \emptyset$ is the relation 1 if $R \neq \emptyset$ and 0 otherwise.

Let R be a logical relation of rank k and $t \in \{0, 1\}^k$ be an assignment. We say that t is *j-compatible*[1] *with R*, if for every subset $T \subset \{1, \ldots, k\}$ of size $|T| < j$ we have $t \mid T \in R \mid T$. Assignment λ is always 0-compatible with the empty relation.

The notion of j-compatibility is the key to characterize weakly antimonotone logical relations.

Lemma 15. *Let R be a logical relation of rank k. The following conditions are equivalent*:

(a) *R is weakly antimonotone.*

(b) *For every $T \subseteq \{1, \ldots, k\}$ and every assignment $t \in \{0, 1\}^{|T|}$ not in $R \mid T$ and $|T|$-compatible with $R \mid T$*:
   (i) $|t|_0 = 0$, *or*
   (ii) $|t|_0 = 1$ *and* $|t|_1 \leq 1$.

(c) *For every $T \subseteq \{1, \ldots, k\}$ and every assignment $t \in \{0, 1\}^{|T|}$ not in $R \mid T$ and $|T|$-compatible with $R \mid T$*:
   (i) $|t|_0 \leq 1$, *and*
   (ii) *If* $|t|_0 = 1$ *then* $|t|_1 \leq 1$.

Proof: The equivalence between conditions (b) and (c) is immediate. We only need to show the equivalence between conditions (a) and (b).

- **[b $\Rightarrow$ a].** For every T, set of positions $0 \leq i_1 < \cdots < i_j \leq k$ and for every assignment $t \in \{0, 1\}^{|T|}$ not in $R \mid T$ and $|T|$-compatible with $R \mid T$ we define

$$C_t^T = \bigvee_{l=1}^{j} x_{i_l}^{t[l]}$$

where

$$x_i^j = \begin{cases} x_i & \text{if } j = 0 \\ \bar{x}_i & \text{otherwise} \end{cases}$$

If t satisfies (b(i)) then C_t^T is antimonotone and falls in type (i) of weakly antimonotone clauses, otherwise t satisfies (b(ii)), C_t^T has at most two literals with at most one affirmed literal and falls in type (ii) of weakly antimonotone clauses. Therefore, C_t^T is a weakly antimonotone clause. From the construction of C_t^T and the assumption $t \notin R \mid T$ it is clear that for all assignments $t' \in \{0, 1\}^k$ that falsify C_t^T we have $t' \notin R$.

For every logical relation R of rank k we define F as the conjunction of all clauses C_t^T where $T \subseteq \{1, \ldots, k\}$ and $t \notin R \mid T$ is a $|T|$-compatible with $R \mid T$ assignment. We show that $[F] = R$:

— It is clear that if $F(t) = 0$ then t falsifies some clause $C_{t'}^T$ and therefore $t \notin R$.
— For the opposite fix an assignment $t \notin R$. Then apply the following algorithm:

Step 1. Assign $i := 1$, $t_1 := t$, $T_1 := \{1, \ldots, k\}$;
Step 2. If t_i is $|T_i|$-compatible with $R \mid T_i$ then *stop*. Otherwise there exists some subset $T \subset T_i$ such that $t_i \mid T \not\in R \mid T$. Then assign $t_{i+1} := t_i \mid T$, $T_{i+1} := T$, $i := i + 1$; go to step 2.

Note that $|T_i|$ decreases in each step of the algorithm. When $T_i = \emptyset$, $t_i = \lambda$ is 0-compatible with the empty relation and therefore the stopping condition is always reached and the algorithm always finds a t_i which is $|T_i|$-compatible with $R \mid T_i$. So, F contains $C_{t_i}^{T_i}$ that is falsified by t.

- **[a $\Rightarrow$ b].** If R is a weakly antimonotone logical relation then for every $T \subseteq \{1, \ldots, k\}$, by Lemma 7, $R \mid T$ is a weakly antimonotone logical relation. Let $F = \bigwedge_{j=1}^{s} C_j$ be a CNF formula, where each clause is weakly antimonotone such that $[F] = R \mid T$. Let $t \not\in R \mid T$ be a $|T|$-compatible with $R \mid T$ assignment. By the $|T|$-compatibility, t cannot falsify any clause of less than $|T|$ literals, and therefore must falsify a clause of exactly $|T|$ literals C_l. If C_l is of type (i) or (ii) in the definition of weakly antimonotone clauses then t satisfies the conditions (b(i)) or (b(ii)) respectively. □

There exists a obvious dual characterization of the weakly monotone logical relations.

3.1.3. Main result in logic. We are now ready to state the main result in logic that we need. It says that every non-basic set of logical relations can express an implication $[\bar{x} \vee \bar{y} \vee z]$ or its dual logical relation $[x \vee y \vee \bar{z}]$.

Theorem 16. *Let S be a finite non-basic set of logical relations, then* $\{[x \vee y \vee \bar{z}], [\bar{x} \vee \bar{y} \vee z]\} \cap \exists\text{-}Relation(S) \neq \emptyset$.

The remainder of this section is devoted to the proof of Theorem 16. The following results are from Schaefer (1978).

Lemma 17 (Schaefer, 1978). *Let R be a logical relation which is not horn, then* $\{[x \not\equiv y], [x \vee y]\} \cap \exists\text{-}Relation(\{R\}) \neq \emptyset$.

Lemma 18 (Schaefer, 1978). *Let R be a logical relation which is not antihorn, then* $\{[x \not\equiv y], [\bar{x} \vee \bar{y}]\} \cap \exists\text{-}Relation(\{R\}) \neq \emptyset$.

Corollary 19 (Schaefer, 1978). *Let S be a finite set of relations which is not horn and not antihorn, then* $[x \not\equiv y] \in \exists\text{-}Relation(S)$.

Lemma 20 (Schaefer, 1978). *Let S be a finite set of logical relations which is not affine and not bijunctive, then* $\exists$-*Relation* $(S \cup \{[x \not\equiv y]\})$ *is the set of all logical relations.*

The following lemmas will apply when condition (c(i)) or (c(ii)) in Lemma 15 fails.

Lemma 21. *Let R be a logical function of rank* $k \geq 2$. *Suppose that there is a k-compatible with R assignment* $t \not\in R$ *that contains at least two zeroes. Then* $\{[x \not\equiv y], [x \vee y]\} \cap Relation(\{R\}) \neq \emptyset$.

Proof: Let $V' = \{x_1, \dots, x_k\}$ be a set of k variables. Let $1 \le i < j \le k$ such that $t[i] = t[j] = 0$. The assignment obtained by flipping either one of the bits i, j in t belongs to R by the k-compatibility of t. Then $[R(x_1, \dots, x_k)[{}^{x_l \in V' - \{x_i, x_j\}}_{t[x_l]}]] \in \{[x \not\equiv y], [x \vee y]\}$. □

Lemma 22. *Let R be a logical relation of rank k* ≥ 3. *Suppose that there is a k-compatible with R assignment t* $\notin$ *R that contains exactly one* 0. *Then* $\{[\bar{x} \vee \bar{y} \vee z], [x \not\equiv y], [x \vee y]\} \cap$ $\exists$-*Relation*$(\{R\}) \neq \emptyset$.

Proof: For every integer $0 < i \le k$, for every assignment $u \in \{0, 1\}^k$, and for every constant $b \in \{0, 1\}$ we define $u_{j \leftarrow b}$ as the assignment obtained from u replacing the jth element by b.

Let $V' = \{x_1, \dots, x_k\}$ be a set of k variables. Let $x_i \in V'$ be the variable such that $t[i] = 0$. By the k-compatibility of t we have $1^k \in R$ and $t_{j \leftarrow 0} \in R$ for all $j \neq i$.

For all $t' \in \{0, 1\}^k$ such that $t' \neq t$ and $t'[i] = 0$, t' can be expressed as $t' = \bigwedge_{j \neq i, t'[j] = 0} t_{j \leftarrow 0}$, so we can assume $t' \in R$, otherwise by Lemma 14 R is not horn and by Lemma 17 $\{[x \not\equiv y], [x \vee y]\} \cap \exists$-Relation$(\{R\}) \neq \emptyset$.

At this point, we can show that a simple implication (i.e., $[x \rightarrow y]$) can be generated from R. We study two cases: If there is no $t' \in R$ such that $t' \neq 1^k$ and $t'[i] = 1$ then let $1 \le j \neq i \le k$ be any integer. Let x_l be any variable in $V - \{x_i, x_j\}$, then $(\exists x_l R(x_1, \dots, x_k)[{}^{V' - \{x_i, x_j\}}_{x_l}]) \equiv (\bar{x}_i \vee x_j)$. Otherwise, let $t' \in R$ be an assignment such that $t' \neq 1^k$ and $t'[i] = 1$, let $W \subset V'$ be the set of variables $x_j \neq x_i$ such that $t'[j] = 1$, and let $x_l \in V - \{x_i\}$ be any variable different from x_i. Then $(R(x_1, \dots, x_k)[{}^{W}_{1}, {}^{V' - W - \{x_i\}}_{x_l}]) \equiv (\bar{x}_l \vee x_i)$.

The double implication (i.e., $[x \wedge y \rightarrow z]$) follows immediately: Let $x_j \in V'$ be a variable not equal to x_i and let $x_l, x_m \in V - \{x_i, x_j\}$. Then, we have $(\exists x_i R(x_1, \dots, x_k) \wedge (\bar{x}_i \vee x_m)[{}^{V' - \{x_i, x_j\}}_{x_l}]) \equiv (\bar{x}_j \vee \bar{x}_l \vee x_m)$. □

The following result follows from Lemmas 21 and 22.

Lemma 23. *Let R be a logical relation of rank k that is not weakly antimonotone. Then* $\{[x \not\equiv y], [\bar{x} \vee \bar{y} \vee z], [x \vee y]\} \cap \exists$-*Relation*$(\{R\}) \neq \emptyset$.

Proof: Let $T \subseteq \{1, \dots, k\}$ and $t \notin R \mid T$ be an $|T|$-compatible with $R \mid T$ assignment such that condition (c) in Lemma 15 is falsified. There are two cases according to what condition is falsified. If condition (c(i)) is falsified then t contains at least two zeros and by Lemma 21 $\{[x \not\equiv y], [x \vee y]\} \cap \exists$-Relation$(\{R \mid T\}) \neq \emptyset$. If condition (c(ii)) is falsified then by Lemma 22 $\{[x \not\equiv y], [\bar{x} \vee \bar{y} \vee z], [x \vee y]\} \cap \exists$-Relation$(\{R \mid T\}) \neq \emptyset$. □

By duality, we have:

Lemma 24. *Let R be a logical relation of rank k that is not weakly monotone. Then* $\{[x \not\equiv y], [x \vee y \vee \bar{z}], [\bar{x} \vee \bar{y}]\} \cap \exists$-*Relation*$(\{R\}) \neq \emptyset$.

Directly from Lemmas 23 and 24 we have:

Corollary 25. *Let S be a finite set of logical relations that is not weakly monotone and not weakly antimonotone, then* $\{[x \not\equiv y], [\bar{x} \vee \bar{y} \vee z], [x \vee y \vee \bar{z}]\} \cap \exists$-*Relation*$(S) \neq \emptyset$.

And now Theorem 16 follows from Corollary 25 and Lemma 20.

3.2. Basic classes are polynomially exactly learnable

In this section, we show efficient learnability results for the basic classes. We use the closure under quantification shown in Section 3.1.1 to prove that in a certain sense each of these classes is embedded in a known learnable class and therefore learnable using the algorithm for the more general class. We start by a formal definition of embedding between classes and a simple observation:

Definition 26. Let $\mathcal{C}, \mathcal{C}'$ be representations of concepts. We say that $\mathcal{C}$ is *embedded* in $\mathcal{C}'$ if there exists a polynomial p such that for every concept name $u \in X$ there exists some concept name $u' \in X$ such that $|u'| \leq p(|u|)$ and $K_{\mathcal{C}}(u) = K_{\mathcal{C}'}(u')$.

Observation 27. Let $\mathcal{C}$ be a representation of concepts which is polynomially exactly learnable with equivalence queries in $\mathcal{H}$. For every representation of concepts $\mathcal{C}'$ embedded in $\mathcal{C}$, $\mathcal{C}'$ is polynomially exactly learnable with equivalence queries in $\mathcal{H}$.

Through the rest of this paper, n denotes the number of variables. We only need a list of learnable classes and to prove that every basic class is embedded in any of them. The only learnable classes that we need are:

Theorem 28 (Angluin, 1988; Valiant, 1984). *For all integers $k \geq 0$ the class $\mathcal{C}_{k-CNF}$ is polynomially exactly learnable with $O(n^k)$ proper equivalence queries.*

Theorem 29 (Chen & Homer, 1997). *Let be $\mathcal{C}_{Af}$ be the set of formulas formed by conjunctions of equations over the two-element field $\{0, 1\}$. The class $\mathcal{C}_{Af}$ is polynomially exactly learnable with $n + 1$ proper equivalence queries.*

These two classes satisfy another nice property: all elements in these classes have size polynomial in n. Specifically:

- For every concept name u in $\mathcal{C}_{k-CNF}$, $|u| \in O(n^k \log n)$. (The $\log n$ factor appears because writing down a variable name requires $\log n$ bits).
- For every concept name u in $\mathcal{C}_{Af}$, $|u| \in O(n^2 \log n)$.

If all the concepts in a representation class $\mathcal{C}$ have size polynomial in n then we do not need to worry about the length of the representation in order to show that any representation class is embedded in $\mathcal{C}$, since this condition is satisfied automatically. From this observation and from Lemma 7 we can derive the following results:

Claim 30. *Let S be a finite set of logical relations. The following conditions hold*:
- *If S is a bijunctive set of logical relations then $\mathcal{C}_{\exists\forall\text{-}Formula(S)}$ is embedded in $\mathcal{C}_{2-CNF}$.*
- *If S is a weakly monotone of degree k set of logical relations then $\mathcal{C}_{\exists\forall\text{-}Formula(S)}$ is embedded in $\mathcal{C}_{k-CNF}$.*

- *If S is a weakly antimonotone of degree k set of logical relations then $\mathcal{C}_{\exists\forall\text{-}Formula(S)}$ is embedded in $\mathcal{C}_{k-CNF}$.*
- *If S is an affine set of logical relations then $\mathcal{C}_{\exists\forall\text{-}Formula(S)}$ is embedded in $\mathcal{C}_{Af}$.*

We are now ready for the positive learnability results. From the previous claim and Observation 27 we can derive the following theorem:

Theorem 31. *Let S be a finite set of logical relations. The following conditions hold*:

- *If S is a bijunctive set of logical relations then $\mathcal{C}_{\exists\forall\text{-}Formula(S)}$ is polynomially exactly learnable with $O(n^2)$ equivalence queries in $\mathcal{C}_{2-CNF}$.*
- *If S is a weakly monotone of degree k set of logical relations then $\mathcal{C}_{\exists\forall\text{-}Formula(S)}$ is polynomially exactly learnable with $O(n^k)$ equivalence queries in $\mathcal{C}_{k-CNF}$.*
- *If S is a weakly antimonotone of degree k set of logical relations then $\mathcal{C}_{\exists\forall\text{-}Formula(S)}$ is polynomially exactly learnable with $O(n^k)$ equivalence queries in $\mathcal{C}_{k-CNF}$.*
- *If S is an affine set of logical relations then $\mathcal{C}_{\exists\forall\text{-}Formula(S)}$ is polynomially exactly learnable with $n + 1$ equivalence queries in $\mathcal{C}_{Af}$.*

3.3. Boolean circuits are pwm-reducible to non-basic classes

In this section, we show that, in a sense made precise later, quantified boolean formulas from non-basic sets of relations are not learnable. We showed in Section 3.1.3 that non-basic sets can express the double implication $[(x \wedge y) \rightarrow z]$ or its dual counterpart. In this section, we complete the proof by showing that sets of quantified formulas containing the double implication can simulate boolean circuits which are known to be not learnable. Simulation in the model of PAC-prediction with membership queries is characterizated by the notion of *pwm-reduction*, as defined above.

First, let us introduce the class of boolean circuits and the non-learnability result that we will use as the basis of our reasoning. Let $\mathcal{C}_{BC}$ be the class of boolean circuits with $\{\vee, \wedge, \neg\}$ gates and $\mathcal{C}_{MBC}$ the class of monotone boolean circuits, i.e., with no '$\neg$' gates. Gates are denoted by natural numbers.

We consider the input variables as gates of fan-in 0. In Angluin & Kharitonov (1995), it is shown that under some cryptographic assumptions boolean circuits are not polynomially predictable with membership queries:

Theorem 32 (Angluin & Kharitonov, 1995). *If there exist public key encryption systems secure against chosen ciphertext attack, then $\mathcal{C}_{BC}$ is not polynomially predictable with membership queries.*

The pwm-reduction from boolean circuits to quantified boolean formulas with non-basic bases is divided in two stages. First, we prove that general boolean circuits are pwm-reducible to monotone boolean circuits and then we show a pwm-reduction from monotone boolean circuits to sets of quantified boolean formulas containing the implication.

Lemma 33. $\mathcal{C}_{BC} \leq_{pwm} \mathcal{C}_{MBC}$.

Proof: Let C be a boolean circuit of size m with $1, \ldots, n$ as input gates. We can assume that $\neg$-gates have in-going edges only from input gates, otherwise by using De Morgan's law repeatedly, we can move $\neg$-gates towards the input gates.

Consider the circuit C' with $1, \ldots, n, m+1, \ldots, m+n$ as input gates and containing as main components subcircuits C_1 and C_2, defined as follows:

Subcircuit C_1 is obtained modifying slightly circuit C. Replace every $\neg$-gate with antecesor i with the new input gate $m+i$.

Circuit C_2 receives as input $1, \ldots, n, m+1, \ldots, m+n$ and the output gate 0 of circuit C_1. C_2 evaluates the following function:

$$\varphi(v_0, v_1, \ldots, v_n, v_{m+1}, \ldots, v_{m+n}) = \begin{cases} v_0 & \text{if } \forall i : 1 \leq i \leq n,\ v_i \neq v_{m+i} \\ 0 & \text{if } \exists i : 1 \leq i \leq n,\ v_i = v_{m+i} = 0 \\ 1 & \text{otherwise} \end{cases}$$

where v_i denotes the value of gate i.

It is easy to construct a monotone boolean circuit evaluating function φ with size polynomial in n.

For all natural number s and for every concept representation $u \in \mathcal{C}_{BC}$ such that $|u| \leq s$, let C be the boolean circuit with n input gates represented by u, let u' be the representation of the monotone boolean circuit C' obtained from C as described above. We define $g(s, n, u) = u'$. For every assignments x and y of length n we define $f(s, n, x) = x\bar{x}$ and

$$h(s, n, xy) = \begin{cases} x & \text{if } x = \bar{y} \\ \perp & \text{if } \exists i : 1 \leq i \leq n, x_i = y_i = 0 \\ \top & \text{otherwise} \end{cases}$$

Clearly, f, g and h satisfy the conditions (1), (2) and (3) in Definition 2 and therefore define a pwm-reduction.

Technical note: In the proof of this prediction with membership reduction and in the next one, functions f, g, h have been defined only partially to keep the proof clear. It is trivial to extend the definition to obtain complete function preserving conditions (1), (2) and (3). □

Lemma 34. $\mathcal{C}_{MBC} \leq_{pwm}$ *∃-Formula*$(\{\bar{x} \vee \bar{y} \vee z\})$.

Proof: Let C be a monotone boolean circuit where m is its number of gates. We can represent each gate j in C by a variable x_j and construct the following formula:

$$F = \exists x_{i_1} \ldots \exists x_{i_r} \varphi \begin{bmatrix} x_o \\ 0 \end{bmatrix}$$

where o is the output gate, $\{i_j : 1 \leq j \leq r\}$ is the set that contains exactly all the $\wedge$-gates and $\vee$-gates except the output gate o, and φ is a conjunction that contains exactly the following clauses:

- For each $\wedge$-gate i, φ contains the clause $\overline{x_j} \vee \overline{x_k} \vee x_i$, where j and k are the ancestors of i.

- For each $\vee$-gate i, φ contains the clauses $\overline{x_j} \vee \overline{x_j} \vee x_i$ and $\overline{x_k} \vee \overline{x_k} \vee x_i$, where j and k are the ancestors of i.

It is easy to show that C and F denote complementary boolean functions.

For all natural number s and for every concept representation $u \in \mathcal{C}_{MBC}$ such that $|u| \leq s$, let C be the monotone boolean circuit with n input gates represented by u, let u' be the representation of the boolean formula F formed from C as shown before. We define $g(s, n, u) = u'$. For all assignment x of length n we define $f(s, n, x) = h(s, n, x) = x$. Clearly, f, g and h satisfy the conditions (1), (2) and (3) in the negated form of Definition 2 and therefore define a pwm-reduction. □

This reduction is very similar to the reduction from the evaluation of monotone boolean circuit problem to the horn satisfiability problem, used to show P-completeness of the latter problem (see, for example, Greenlaw, Hoover, & Ruzzo, 1995).

By duality we have:

Lemma 35. $\mathcal{C}_{MBC} \leq_{pwm} \mathcal{C}_{\exists\text{-}Formula(\{x \vee y \vee \bar{z}\})}$.

We put the previous lemmas together with Theorems 16 and 32 and we obtain the following result:

Corollary 36. *Let S be a finite non-basic set of logical relations, then*:
(a) *The set* $\exists$-*Formula*(S) *contains* $[x \vee y \vee \bar{z}]$ *or* $[\bar{x} \vee \bar{y} \vee z]$.
(b) *The class* $\mathcal{C}_{MBC}$ *is pwm-reducible to* $\mathcal{C}_{\exists\text{-}Formula(S)}$.
(c) *The class* $\mathcal{C}_{BC}$ *is pwm-reducible to* $\mathcal{C}_{\exists\text{-}Formula(S)}$.
(d) *The class* $\mathcal{C}_{\exists\text{-}Formula(S)}$ *is not polynomially predictable with membership queries under the assumption that public key encryption systems secure against CC-attack exist.*

Finally, Theorem 6 follows from Theorem 31 and Corollary 36.

Acknowledgments

I am very grateful to Ricard Gavaldà for very helpful discussions and extensive comments on this work and its presentation, and for pointing out the relationship given in Lemma 34. This work was supported in part by the Spanish Government through grant AP94 43716784 and the DGICYT (project KOALA BP95-0797) and the EC through the Esprit BRA Program (Working Group 8556, NeuroColt and project 20244, ALCOM IT).

Note

1. The notion of compatibility used in this paper differs only slightly from the compatibility defined by Kavvadias & Sidderi (1996). Precisely, the j-compatibility corresponds exactly to the $(j - 1)$-compatibility in the sense of Kavvadias & Sidderi (1996).

References

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

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

Angluin, D., & Kharitonov, M. (1995). When won't membership queries help? *Journal of Computer and System Sciences, 50*, 336–355.

Berggren, U. (1993). Linear time deterministic learning of k-term DNF. *Proc. 6th Annual Workshop on Computational Learning Theory* (pp. 37–40).

Chen, Z., & Homer, S. (1997). Learning counting functions with queries. *Theoretical Computer Science, 180*, 155–168.

Creignou, N. (1995). A dichotomy theorem for maximum generalized satisfiability problems. *Journal of Computer and System Sciences, 51*(3), 511–522.

Creignou, N., & Hermann, M. (1996). Complexity of generalized satisfiability counting problems. *Information and Computation, 125,* 1–12.

Dalmau, V. (1997). *Some dichotomy theorems on constant-free quantified boolean formulas* (Technical Report LSI-97-43-R), Department LSI, Universitat Politécnica de Catalunya.

Dor, D., & Tarsi, M. (1997). Graph decomposition is NP-complete: A complete proof of Holyer's conjecture. *SIAM Journal on Computing, 26*(4), 1166–1187.

Fortune, S., Hopcroft, J., & Wyllie, J. (1980). The directed subgraph homeomorphism problem. *Theor. Comp. Sci., 81*(2), 111–121.

Greenlaw, R., Hoover, H.J., & Ruzzo, W.L. (1995). *Limits to Parallel Computation: P-completeness theory*. Oxford University Press.

Hell, P., & Nešetřil, N. (1990). On the complexity of H-coloring. *J. Combin. Theory Ser. B, 48*, 92–110.

Kavvadias, D., & Sidderi, M. (1996). The inverse satisfiability problem. *Proc. 2nd Annual International Conference on Computing and Combinatorics*, COCOON'96 (pp. 250–259).

Kearns, M., & Valiant, L. (1994). Cryptographic limitations on learning boolean formulae and finite automata. *J. ACM, 41*(1), 67–95.

Khanna, S., Sudan, M., & Trevisan, L. (1997). Constraint satisfaction: The approximability of minimization problems. *Proc. of the 12th IEEE Conference on Computational Complexity*.

Khanna, S., Sudan, M., & Williamson, P. (1997). A complete classification of the approximability of maximization problems derived from boolean constraint satisfaction. *Proc. of the 29th Annual ACM Symposium on the Theory of Computing*.

Maass, W., & Turán, G. (1995). On learnability and predicate logic. *Proc. Bar-Ilan Symposium on the Foundations of Artificial Intelligence*, BISFAI'95 (pp. 75–85).

Papadimitriou, C.H. (1994). *Computational Complexity*. Addison-Wesley.

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

Schaefer, T.J. (1978). The complexity of satisfiability problems. *Proc. 10th Annual ACM Symposium on Theory of Computing* (pp. 216–226).

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

Received July 23, 1997
Accepted October 16, 1998
Final manuscript December 14, 1998