Some Dichotomy Theorems on Constant-free Quantified Boolean Formulas

Víctor Dalmau*

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

10 September 1997

Abstract

In this paper we study the satisfiability of constant-free quantified boolean formulas. We consider the following classes of quantified boolean formulas. Fix a finite set of basic boolean logical functions. Take conjunctions of these basic functions applied to variables in arbitrary way. Finally, quantify existentially or universally some of the variables.

Schaefer [20] earlier studied the satisfiability of quantified boolean formulas with constants. He showed that every such problem is either in P or PSPACE-complete and he gave a complete classification of the tractable cases. We extend the PSPACE-hardness results to constant-free quantified boolean formulas obtaining a dichotomy theorem for the satisfiability of constant-free quantified boolean formulas. We find that, in fact, constants do not make a difference when considering the satisfiability of quantified boolean formulas. We also prove a dichotomy theorem that allows us to improve a previous result on the learnability of quantified boolean formulas getting rid of the constants.

*Work 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).

1 Introduction

The problem of determining whether a given propositional formula is satisfiable has been widely studied. It is well known that even restricted to formulas in conjunctive normal form with at most three literals per clause the problem is NP-complete [5]. Therefore, researchers have attempted to determine the satisfiability of subclasses of propositional boolean formulas, in particular inside CNF. For example, the satisfiability in polynomial time of bijunctive formulas was shown in [5] and can be done also in linear time [17]. The satisfiability of Horn formulas and read-twice CNF formulas have also been shown to be decidable in linear time [9, 18].

Schaefer [20] proposed a large set of classes of boolean formulas and studied their associated satisfiability problem, called the *Generalized Satisfiability Problem*. He proved a very remarkable result: For every such class of formulas, its satisfiability is either in P or NP-complete. This dichotomy is somewhat surprising since one might expect that any such large and diverse class of problems would include some representative of the many intermediate degrees of complexity which presumably lie between these two extremes. Furthermore, he gave a complete classification of the polynomial-time decidable classes.

We now provide the informal definitions to state Schaefer's classification result for the satisfiability problem. Precise definitions of the concepts involved will be given in Section 2. Let $S = \{R_1, \ldots, R_m\}$ be a finite set of logical relations. Define a Formula(S) to be any boolean formula formed by conjunctions of any number of clauses of the form $R_i(x_1, \ldots, x_k)$, where $x_1, \ldots, x_k$ are possibly repeated variables and k is the rank of R_i.

An example: Consider the set of boolean formulas formed by a conjunction of clauses with at most three literals per clause. Every such formula can be expressed as a formula in Formula(S) with the set of logical relations $S = \{R_0, R_1, R_2, R_3\}$, defined by:

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

Schaefer showed that the satisfiability of Formula(S) is polynomial-time decidable if and only if at least one of the following condition holds:

- Every relation in S is satisfied when all variables are 0.
- Every relation in S is satisfied when all variables are 1.

- Every relation in S is definable by a CNF formula in which each clause has at most 2 literals.
- Every relation in S is definable by a CNF formula in which each clause has at most one negated variable.
- Every relation in S is definable by a CNF formula in which each clause has at most one unnegated variable.
- Every relation in S is the set of solutions of a system of linear equations over the two-element field $\{0, 1\}$.

We are interested in the satisfiability of quantified boolean formulas. It is well known that this problem is PSPACE-complete [21, 22] and some effort has been devoted to classify the polynomial-time decidable classes. A linear time algorithm when all clauses have at most two variables can be found in [3]. In [13] a cubic time algorithm for quantified Horn formulas has been showed.

We define an $\exists\forall$-Formula(S) to be the set of quantified boolean formulas that we obtain if we take a formula in Formula(S) and we quantify existentially or universally in any order some of the variables. We call the formulas constructed in this way *Generalized quantified boolean formulas.* If we allow to put constants $\{0, 1\}$ in the place of variables then we have a larger set of formulas denoted by $\exists\forall$-Formula$_C(S)$.

Schaefer [20] showed a dichotomy theorem for quantified boolean formulas with constants. More precisely, he proved that the satisfiability of $\exists\forall$-Formula$_C(S)$ (with constants) is polynomial time decidable if and only if at least one of the following conditions holds:

- Every relation in S is definable by a CNF formula in which each clause has at most 2 literals.
- Every relation in S is definable by a CNF formula in which each clause has at most one negated variable.
- Every relation in S is definable by a CNF formula in which each clause has at most one unnegated variable.
- Every relation in S is the set of solutions of a system of linear equations over the two-element field $\{0, 1\}$.

The use of formulas with constants instead of the most natural constant-free formulas is a technical trick that is been used as an intermediate step in the quantifier-free case because it simplifies the proof for the NP-hard cases. For the quantifier-free case Schaefer [20] gave a way to keep the result free of constants. In fact, he found two classes of logical relations, namely the 0-valid and 1-valid logical relations, that are polynomial-time decidable in the non-constant case and such that if a set of logical relations does not belong to any of both classes then it is possible to construct boolean formulas that simulate the constants. Nevertheless, such result is not proved for the quantified formulas and the analogous dichotomy for quantified formulas without constants remained open.

From the previous result it is not clear what happens when the constants are removed. It is obvious that the positive results remain true even without constants because the problem becomes easier, but for the negative results one such easy generalization is not possible. Does the problem remain PSPACE-complete even if we do not allow constants, or some new classes appear that are PSPACE-complete if we allow constants and become polynomial-time decidable in the constant-free case.

We give an answer to this question in this paper. More precisely, we find some sets of logical relations that play, for the quantifier free case, the role of the 0-valid and 1-valid relations. These sets are embedded in the polynomial time decidable classes and, therefore, in contrast to the case of non-quantified boolean formulas, the negative results remain unchanged; see beginning of Section 3 for formal statement.

The main aim of this paper is to study the complexity of deciding the satisfiability of quantified boolean formulas. However, some intermediate results that we need for the proofs have interesting applications. Using this intermediate results, we also prove a dichotomy theorem that gets rid of constants a previous result on the learnability of quantified boolean formulas that we proved in [8]; see Section 4.

A few but not many dichotomy results in complexity theory are already known. The first one is the dichotomy result for the generalized satisfiability decision problem by Schaefer [20]. The others concern the H-coloring of graphs [11], the subgraph homeomorphism [10], the inverse generalized satisfiability problem [14], the generalized satisfiability counting problem [7], the aproximability of minimization and maximization problems [6, 15, 16], and the learnability of quantified boolean formulas [8]. It is remarkable that most of the dichotomy results shown before are in the framework of generalized satisfiability problems proposed by Schaefer. We hope that our work will be a first step to extend and generalize these results to the quantified boolean formulas.

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. Let X denote $\{0,1\}^*$. 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. A logical relation of rank k (k integer) is a subset of $\{0,1\}^k$. There exists a unique assignment of length 0. We call it λ.

We use the term *formula* in a wide 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 *constant-free boolean formulas over the basis S*, denoted by $\text{Formula}(S)$, is the smallest set of formulas such that:

- For all $R \in S$ of rank k, $R(x_{i_1}, \ldots, x_{i_k}) \in \text{Formula}(S)$ where $x_{i_j} \in V$ for $1 \leq j \leq k$.
- For all $F, G \in \text{Formula}(S)$, $F \wedge G \in \text{Formula}(S)$.

The set of *boolean formulas with constants over the basis S* denoted by $\text{Formula}_C(S)$, is the smallest set of formulas such that:

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

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

- For all $F \in \text{Formula}(S)$, $F \in \exists\forall\text{-Formula}(S)$.
- For all $F \in \exists\forall\text{-Formula}(S)$ and for all $x \in V$, $\exists x F$ and $\forall x F$ are in $\exists\forall\text{-Formula}(S)$.

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

- For all $F \in \text{Formula}_C(S)$, $F \in \exists\forall\text{-Formula}_C(S)$.
- For all $F \in \exists\forall\text{-Formula}_C(S)$ and for all $x \in V$, $\exists x F$ and $\forall x F$ are in $\exists\forall\text{-Formula}_C(S)$.

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

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. Let F be a formula over V, let $v, w \in V \cup \{0,1\}$ be variables or constants. We define $F\left[\begin{smallmatrix} v \\ w \end{smallmatrix}\right]$ as the formula formed from F by replacing each occurrence of v by w. If $W \subseteq V$ is a set of variables then $F\left[\begin{smallmatrix} W \\ w \end{smallmatrix}\right]$ denotes the result of substituting w for every occurrence of every variable in W. Multiple substitutions are denoted by expressions such as $F\left[\begin{smallmatrix} V \\ w \end{smallmatrix}, \begin{smallmatrix} V' \\ w' \end{smallmatrix}, \begin{smallmatrix} V'' \\ w'' \end{smallmatrix}\right]$ with obvious meaning. For every set of logical functions S we define Relation$(S) = \{[F] : F \in \text{Formula}(S)\}$. Similarly, we define $\exists\forall$-Relation(S), $\exists$-Relation(S), Relation$_C(S)$, $\exists\forall$-Relation$_C(S)$, and $\exists$-Relation$_C(S)$. We use the term *clause* in a broad sense to mean any boolean relation in a conjunction. A clause is *bijunctive* if it is the disjunction of at most 2 literals. A clause is *horn* (resp. *antihorn*) if it is the disjunction of at most one unnegated (resp. negated) variables. A clause is *monotone* (resp. *antimonotone*) if it is the disjunction of unnegated variables (resp. negated variables). A clause is *weakly monotone* (resp. *weakly antimonotone*) if it is the disjunction of unnegated (resp. negated) variables or the disjunction of at most two literals with at most one negated (resp. unnegated) variable. A clause is *equality monotone* (resp. *equality antimonotone*) if it is the equality between two variables or it is the disjunction of unnegated variables (resp. negated variables). A clause is *affine* if it is an equation over the two-element field $\{0,1\}$. A clause of rank k is *0-valid* (resp. *1-valid*) if it is empty or it is satisfied by 0^k (resp. 1^k). A clause is *0-affine* if it is affine and 0-valid. A clause is *1-affine* if it is affine and 1-valid.

We can extend the definitions above to relations, formulas, and sets of relations: The logical relation R of rank k is bijunctive if $R(x_1, \ldots, x_k)$ is logically equivalent to some conjunctive formula where each clause is bijunctive. The formula F is bijunctive if $[F]$ is bijunctive. The set S of logical relations is bijunctive if every $R \in S$ is bijunctive.

Analogous definitions can be done if we substitute bijunctive by horn, antihorn, monotone, antimonotone, weakly monotone, weakly antimonotone, equality monotone, equality antimonotone, affine, 0-valid, 1-valid, 0-affine, or 1-affine.

We assume some familiarity with complexity theoretical notions, in particular, with definitions of complexity classes P, NP, PSPACE, polynomial time reducibility and completeness. We refer the reader to [4, 17].

3 The Satisfiability of Quantified Boolean Formulas

This section is devoted to state and prove the dichotomy theorem for the satisfiability of quantified boolean formulas. Let $\mathcal{F}$ be a set of boolean formulas, SAT($\mathcal{F}$) is the following problem: Given as input a formula $F \in \mathcal{F}$ decide if F is satisfiable.

Theorem 1 *(Dichotomy Theorem for the satisfiability of quantified boolean formulas) Let S be a finite set of logical relations. If S satisfies one of the conditions (a)-(d) below, then SAT($\exists\forall$-Formula(S)) is polynomial-time decidable. Otherwise, SAT($\exists\forall$-Formula(S)) is log-complete in PSPACE.*

a.- S is bijunctive.

b.- S is affine.

c.- S is horn.

d.- S is antihorn.

Part of the work is made in the article of Schaefer [20]. In fact, he proved a dichotomy theorem for the quantified boolean formulas with constants.

Theorem 2 *[20](Dichotomy Theorem for the satisfiability quantified boolean formulas with constants) Let S be a finite set of logical relations. If S satisfies one of the conditions (a)-(d) in Theorem 1, then $SAT(\exists\forall\text{-}Formula_C(S))$ is polynomial-time decidable. Otherwise, the problem $SAT(\exists\forall\text{-}Formula_C(S))$ is log-complete in PSPACE.*

There are two remarkable facts to note here: First, when we consider boolean formulas with constants the tractable cases remain unchanged if we allow both universal and existential quantifiers, only existential quantifiers or not quantifiers at all. However, in this two last cases the intractable cases become NP-complete instead of PSPACE-complete.

Second, note that, in contrast to the quantifier free case, the tractable cases are the same with and without constants. Therefore, constants do not matter when we consider the satisfiability of quantified boolean formulas.

Here is a bird's eye of the proof.

We only need to prove that when a set of logical relations S does not fall in the tractable cases then there exists a way to simulate constants. First, we give semantic characterizations of some families of logical relations defined in

Section 2. In a semantic characterization we define a kind of logical relations in function of some property that must be satisfied by all the assignments. This condition is more manegable than the original definition, in terms of expressibility by conjunction of a type of clauses (Due to space restrictions, the semantic characterizations are moved to the Appendix).

Then, we prove that if a set of logical relations S does not belong to some of certain classes of logical relations contained in the tractable cases, namely, the 0-affine, 1-affine, equality monotone, and equality antimonotone then at least one of these conditions holds:

- The logical relations $[x]$ and $[\overline{x}]$ can be generated with quantified formulas over S.
- The logical relation $[x \not\equiv y]$ can be generated with a quantified formula over S and every relation $R \in S$ satisfies the following property: If t satisfies R then $\overline{t}$ also satisfies R.

Finally, we prove that any of the previous conditions is enough to simulate constants.

3.1 Technical Results

A logical relation R of rank k is called *complementive* if for every assignment $t \in \{0,1\}^k$, $t \in R$ if and only if $\overline{t} \in R$. This definition can be extended naturally to formulas, sets of logical relations, and sets of formulas. The sets of complementive logical relations satisfy another nice property: Let S be a complementive set of logical relations, then $\exists\forall$-Relation(S) is also complementive. This property can be easily verified by induction.

From Schaefer's results we have the following theorem:

Theorem 3 *[20] Let S be a set of logical relations. Then at least one of the following conditions holds:*

a.- Every relation in S is 0-valid.

b.- Every relation in S is 1-valid.

c.- The logical relations $[x]$ and $[\overline{x}]$ are contained in $\exists$-Relation(S).

d.- The logical relation $[x \not\equiv y]$ is contained in $\exists$-Relation(S) and every relation in S is complementive.

The usefulness of the previous theorem is the following: if condition (c) or (d) holds then with a simple construction we can simulate constants and in consequence if a set of logical relations S is not 0-valid and not 1-valid then $\text{Relation}_C(S) = \text{Relation}(S)$.

The aim of this section is to state the counterpart of Theorem 3 for generalized quantified boolean formulas. The proof is given in the Appendix.

Theorem 4 *Let S be a set of logical relations. Then at least one of the following conditions holds:*

a.- Every relation in S is 0-affine.

b.- Every relation in S is 1-affine.

c.- Every relation in S is equality antimonotone.

d.- Every relation in S is equality monotone.

e.- The logical relations $[x]$ and $[\overline{x}]$ are contained in $\exists\forall$-Relation(S).

f.- The logical relation $[x \not\equiv y]$ is contained in $\exists\forall$-Relation(S) and every relation in S is complementive.

It must be mentioned that if we substitute the classes appearing in $(a) - (d)$ by more general classes, then Theorem 4 remains true and for obtaining Theorems 1 and 5 we only need a weaker and easier variant constructed in this way. However, we prefer to state the theorem in this way because classes in $(a) - (d)$ are the smallest that satisfy Theorem 4, and therefore this result can be useful to prove other results in the framework of quantified boolean formulas.

3.2 Proof of Theorem 1

With the help of Theorem 4 the proof of Theorem 1 is straightforward. This proof follows the lines of the proof of Theorem 2.1 in [20]. First, note that the tractable cases follow directly from Theorem 2, since we can use the same algorithm for the satisfiability of formulas with constants to decide the satisfiability of formulas without constants.

The PSPACE-hardness result follows from Theorem 4 taking into account that the equality antimonotone (resp. equality monotone) relations are a subset of horn relations (resp. antihorn relations) and that the 0-affine and 1-affine relations are a subset of affine relations. Therefore if conditions

$(a) - (d)$ in Theorem 1 do not hold, then conditions (e) or (f) in Theorem 4 hold. In both cases we can prove the following log-space reduction

$$\mathrm{SAT}(\exists\forall\text{-Formula}_C(S)) \leq_{log} \mathrm{SAT}(\exists\forall\text{-Formula}(S)).$$

Therefore, SAT($\exists\forall$-Formula(S)) is PSPACE-hard if so is the corresponding problem with constants.

If condition (e) holds then we can simulate the constants in every formula F by the construction $G = \exists x_0 \exists x_1 (F \wedge \overline{x_0} \wedge x_1) \left[{0 \atop x_0}, {1 \atop x_1} \right]$. Otherwise, condition (f) holds. Therefore, every formula in $\exists\forall$-Formula(S) is complementive and for every formula $F \in \exists\forall\text{-Formula}_C(S)$, the constant-free formula $G = \exists x_0 \exists x_1 F \wedge (x_0 \not\equiv x_1) \left[{0 \atop x_0}, {1 \atop x_1} \right]$ is satisfiable if and only if F is. In this case we cannot construct the functions 0 and 1, but we can assure that variables x_0 and x_1 take different values. This is enough because the formula G is complementive.

In both cases F is satisfiable if and only G is, and G is logspace computable given F.

■

4 Learning Result

Theorem 4 can be applied to get rid of constants some other results over quantified boolean formulas. In this section, we complete a previous result on the learnability of quantified boolean formulas with constants proved in [8]. Our dichotomy for learnability theorem is stated right now.

Theorem 5 *(Dichotomy Theorem for the learnability of quantified boolean formulas) Let S 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, $\mathcal{C}_{\exists\forall\text{-}Formula(S)}$ 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 the Appendix for the definitions and proof of this theorem.

5 Acknowledgments

I am very grateful to Ricard Gavaldà for very helpful discussions and extensive comments on this work and its presentation. I also like to thank Carlos Domingo for his helpful comments.

References

[1] D. Angluin. *Queries and concept learning*, Machine Learning 2 (1988), 319-342.

[2] D. Angluin, and M. Kharitonov. *When won't membership queries help?*, J. Comput. System Sci. 50 (1995) 336-355.

[3] B. Aspvall, M. Plass, and R. Tarjan. *A linear-time algorithm for testing the truth of certain quantified Boolean formulas*, Inform. Process. Lett. 8 (1979), 121-123.

[4] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity. Vol I*, EATCS Monographs on Theoretical Computer Science, Springer-Verlag, Berlin (1988).

[5] S. Cook. *The Complexity of theorem-proving procedures*, in "Proceedings of the 3th Annual ACM Symposium on Theory of Computing, STOC'71" (1971), 151-158.

[6] N. Creignou. *A Dichotomy Theorem for Maximum Generalized Satisfiability Problems*, Journal of Computer and System Sciences 51(3), (1995) 511-522.

[7] N. Creignou, and M. Hermann. *Complexity of Generalized Satisfiability Counting Problems*, Information and Computation 125, (1996), 1-12.

[8] V. Dalmau. *A Dichotomy Theorem for Learning Quantified Boolean Formulas*, in "Proceedings of the 10th Annual Conference on Computational Learning Theory, COLT'97" (1997), 193-200.

[9] W. Dowling and J. Gallier. *Linear-time algorithms for testing the satisfiability of propositional Horn formulae*, Journal of Logic Programming 1(3) (1984)

[10] S. Fortune, J. Hopcroft, and J. Wyllie. *The directed subgraph homeomorphism problem*, Theor. Comp. Sci. 81(2) (1980), 111-121.

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

[12] R. Karp. *Reducibility among combinatorial problems*, in "Complexity of Computer Computations". Plenum Press (1972) 85-104.

[13] M. Karpinski, M. Kleine Büning, and P. Schmitt. *On the computational complexity of quantified Horn clauses*, in "Proceedings CSL'87" (1987), 129-137.

[14] D. Kavvadias, and M. Sidderi. *The Inverse Satisfiability Problem*, in "Proceedings of the 2nd Annual International Conference on Computing and Combinatorics, COCOON'96" (1996), 250-259.

[15] S. Khanna, M. Sudan, and L. Trevisan. *Constraint Satisfaction: The Approximability of Minimization Problems*, in "Proceedings of the 12th IEEE Conference on Computational Complexity." (1997).

[16] S. Khanna, M. Sudan, and P. Williamson. *A complete classification of the approximability of maximization problems derived from boolean constraint satisfaction*, in "Proceedings of the 29th Annual ACM Symposium on the Theory of Computing". (1997).

[17] C. H. Papadimitriou. *Computational complexity*, Addison-Wesley (1994).

[18] K. Pillaipakkamnatt and V. Raghavan. *A linear time equivalence test for read-twice DNF formulas*, Inf. Process. Let. 48, (1993) 289-295.

[19] L. Pitt, and M. Warmuth. *Prediction preserving reducibility*, J. Comput. System Sci. 41 (1990), 430-467.

[20] T. Schaefer. *The complexity of satisfiability problems*, in "Proceedings of the 10th Annual ACM Symposium on Theory of Computing, STOC'78" (1978), 216-226.

[21] L. Stockmeyer. *The polynomial-time hierarchy*, Theor. Comp. Sci. 3 (1976) 1-22.

[22] L. Stockmeyer and A. Meyer. *Word problems requiring exponential time: preliminary report*, in "Proceedings of the 5th Annual ACM Symp. on Theory of Computing STOC'73" (1973), 1-9.

6 Appendix

6.1 Proof of Theorem 4

Semantic Characterizations

First, we give semantic characterizations of some sets of logical relations. More precisely, we present semantic characterizations for the 0-affine, horn, antimonotone, and equality antimonotone logical relations. Semantic characterizations for the dual classes are not stated but are used implicitly later. Other semantic characterizations can be found in [20, 7].

The following result is known (see [7], for example). It characterizes the class of the 0-affine sets. It shows that 0-affine relations express vectorial subspaces over the two-element field $\{0, 1\}$.

Lemma 6 *[7](Semantic characterization of* 0*-affine relations) Let R be a logical relation of rank k. Then the following statements are equivalent:*

a.- R is 0*-affine.*

b.- i.- The assignment 0^k *belongs to R, and*

ii.- For all assignments $t_1, t_2 \in R$, $t_1 \oplus t_2 \in R$.

At this point we show a characterization of the horn, antimonotone, and equality antimonotone relations. We need the following notation:

For every logical relation R of rank k and for every set $T = \{i_1, \ldots, i_j\}$ of j positions $0 \leq i_1 < \ldots < i_j \leq k$ we define $R|T$ as the subset of $\{0, 1\}^j$ such that $t \in R|T$ if and only if there exists some assignment $t' \in R$ such that t agrees with t' in the positions indexed by T. That is, $R|T = [\exists x_{l_1}, \ldots, \exists x_{l_{k-j}} R(x_1, \ldots, x_k)]$ where $\{l_1, \ldots, l_{k-j}\} = \{1, \ldots, k\} - T$, and therefore $R|T \in \exists\text{-Relation}(\{R\})$.

For all $t \in \{0, 1\}^k$ and $T \subset \{1, \ldots, k\}$ we define $t|T$ as the assignment $t' \in \{0, 1\}^{|T|}$ that agrees with t in the positions indexed by T. We have $t|\emptyset = \lambda$ and $R|\emptyset$ is the relation *true* if $R \neq \emptyset$ and *false* otherwise.

Let R be a logical relation of rank k and $t \in \{0, 1\}^k$ an assignment. We say that t is *j-compatible*[1] if for every subset $T \subset \{1, \ldots, k\}$ of size $|T| < j$ we can find some assignment $t' \in R$ such that t and t' agree in the positions indexed by T. Assignment λ is always 0-compatible.

The notion of j-compatibility is the key to characterize horn logical relations:

[1]The notion of compatibility used in this paper differs only slightly from the compatibility defined by Kavvadias and Sidderi [14]. Precisely, the j-compatibility here is the $(j - 1)$-compatibility in the sense of [14].

Lemma 7 *(Semantic characterization of horn logical relations) Let R be a logical relation of rank k. Then the following statements are equivalent:*

a.- R is horn.

b.- For all $t_1, t_2 \in R$, $t_1 \wedge t_2 \in R$.

c.- For every $T \subseteq \{1, \ldots, k\}$ and for every $|T|$-compatible assignment t not in $R|T$, $|t|_0 \leq 1$.

Proof.
$[(a) \Longleftrightarrow (b)]$ See [17].
$[(a) \Rightarrow (c)]$ If R is horn, then $R|T$ is horn. Let $F = \bigwedge_{j=1}^{s} C_j$ be a CNF formula where each clause is horn such that F denotes the logical relation $R|T$. Let $t \notin R|T$ be a $|T|$-compatible 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. The clause C_l contains at most one unnegated literal, so $|t|_0 \leq 1$.
$[(c) \Rightarrow (a)]$ For every set of positions $T = \{i_1, \ldots, i_{|T|}\}$, $0 \leq i_1 < \ldots < i_{|T|} \leq k$ and for every $|T|$-compatible assignment $t \notin R|T$ we define

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

where

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

By assumption (c), $|t|_0 \leq 1$, so C_t^T is a horn clause. The set of assignments $t' \in \{0,1\}^k$ that coincide with t in T is the set of assignments that falsify C_t^T. Since $t \notin R|T$, for every t' that falsifies 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 $T \subset \{1, \ldots, k\}$ and $t \notin R|T$ is a $|T|$-compatible assignment. We show that $[F] = R$:

- It is clear that if $F(t) = 0$ then t falsifies some clause $C_{t'}^T$ and therefore $t \notin R$.

- Fix an assignment $t \notin R$. We apply the following algorithm:

  Step 1 Assign $t_1 := t$, $T_1 = \{1, \ldots, k\}$, $i := 1$;

Step 2 If t_i is $|T_i|$-compatible with respect to the relation $R|T_i$ then **stop**, otherwise there exists some subset $T \subset T_i$ such that $t_i|T \notin R|T$. Assign $t_{i+1} := t_i|T$, $T_{i+1} := T$, $i := i+1$; 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 and therefore condition in step (2) is always reached and the algorithm always finds a t_i which is $|T_i|$-compatible with respect the relation $R|T_i$. So, F contains $C_{t_i}^{T_i}$ that is falsified by t, and $F(t) = 0$.

■

The technique used to prove the identity of conditions (a) and (c) in the previous lemma can be applied to other kinds of logical relations $\mathcal{R}$. The only assumption we need is that these sets are 'closed' in the following sense: If R belongs to $\mathcal{R}$ then for all T, $R|T$ belongs to $\mathcal{R}$. The set of antimonotone logical relations satisfies this property and therefore we can prove in an identical way the following lemma:

Lemma 8 *(Semantic characterization of antimonotone logical relations) Let R be a logical relation of rank k. Then the following statements are equivalent:*

a.- R is antimonotone.

b.- For all assignments $t_1 \leq t_2$ of length k, $t_2 \in R$ implies $t_1 \in R$.

c.- For every $T \subseteq \{1, \ldots, k\}$ and for every $|T|$-compatible assignment t not in $R|T$, $t = 1^{|T|}$.

For the equality antimonotone logical relations we show that in a certain sense they are antimonotone logical relations applied to sets of variables with the same value:

Lemma 9 *(Semantic characterization of equality antimonotone logical relations) Let R be a logical relation of rank k. Then the following statements are equivalent:*

a.- R is equality antimonotone.

b.- There exists some collection of sets of indices $T_1, \ldots, T_r$ such that:

i.- For all $1 \leq i, j \leq r$, $T_i \cap T_j = \emptyset$ and $\bigcup_{i=1}^{r} T_i = \{1, \ldots, k\}$, and *ii.- For all $t \in R$, $1 \leq i \leq r$, and $j, l \in T_i$, $t[j] = t[l]$, and*

iii.- The boolean formula $R(x_1, \ldots, x_k) \begin{bmatrix} T_1 \\ y_1 \end{bmatrix} \cdots \begin{bmatrix} T_r \\ y_r \end{bmatrix}$ is antimonotone.

Proof.
$[(a) \Rightarrow (b)]$ There exists some boolean formula over the variables $x_1, \ldots, x_k$, $F = \bigwedge_{j=1}^{s} C_j$ where each C_j is an equality between two variables or a disjunction of negated variables such that $[F] = R$. Let $\mathcal{R}$ be a relation over $\{1, \ldots, k\}$ defined in the following way: $i\mathcal{R}j$ if $x_i = x_j$ is a clause in F, and let $\mathcal{R}^*$ be its reflexive, symmetric, and transitive closure. Clearly $\mathcal{R}^*$ is an equivalence relation and therefore defines a partition $T_1, \ldots, T_r$ over the set $\{1, \ldots, k\}$. It is easy to verify that this partition satisfies conditions (bi), (bii), and $(biii)$.
$[(b) \Rightarrow (a)]$ Let $\mathcal{T} = \{T_1, \ldots, T_r\}$ be a collection of sets that satisfies condition (b). Let $F = \wedge_{j=1}^{s} C_j$ be a boolean formula in the variables $y_1, \ldots, y_r$ where each clause is antimonotone and such that we have $[F] = R(x_1, \ldots, x_r) \begin{bmatrix} T_1 \\ y_1 \end{bmatrix} \ldots \begin{bmatrix} T_r \\ y_r \end{bmatrix}$. For every $1 \leq i \leq r$, let $T_i = \{j_{i,1}, \ldots, j_{i,l_i}\}$ where $l_i = |T_i|$. We define F_i in the following way:

$$F_i = (x_{j_{i,1}} = x_{j_{i,2}}) \wedge \cdots \wedge (x_{j_{i,l_i - 1}} = x_{j_{i,l_i}})$$

Then we construct the following formula:

$$G = F \begin{bmatrix} y_1 \\ x_{j_{i,1}} \end{bmatrix} \ldots \begin{bmatrix} y_r \\ x_{j_{r,1}} \end{bmatrix} \wedge \bigwedge_{i=1}^{r} F_r$$

Clearly $[G] = R$ and every clause of G is an equality between two variables or a disjunction of negated variables. Therefore R is equality antimonotone. ∎

Proof of Theorem 4

In the case of quantified boolean formulas the 0-validity (resp. 1-validity) is not in general preserved when we quantify universally. This is the fundamental reason why Theorem 3 does not hold for quantified boolean formulas. The following results show simple conditions that imply that 0-validity is lost.

Lemma 10 *Let R be a logical relation of rank k. If there exist two different assignments $t, t' \in R$ such that $t \leq t'$ and $t \oplus t' \notin R$, then $\exists\forall$-Relation$(\{R\})$ is not 0-valid.*

Proof. Let $T \not\subseteq \{1,\ldots,k\}$ be the proper subset of indices i such that $(t \oplus t')[i] = 1$ and let be x_j a new variable, then $\forall x_j R(x_1,\ldots,x_k) \left[\begin{smallmatrix}\{x_l : l \in T\} \\ x_j\end{smallmatrix}\right]$ contains $t|(\{1,\ldots,k\} - T)$ (i.e. is not empty) but not $0^{(k-|T|)}$, therefore is not 0-valid. ∎

Corollary 11 *Let R be a 1-valid logical relation of rank k. Then the class $\exists\forall$-Relation$(\{R\})$ is complementive or not 0-valid.*

Proof. Assume that R is not complementive. Let t be an assignment such that $t \in R$ and $\bar{t} \notin R$. Apply Lemma 10 taking t and 1^k. ∎

These results are extensively used in later proofs to show the non-0-validity of some sets. First, we assume that we can use a constant $b \in \{0,1\}$. We define Relation$_b(S)$ as the subset of Relation$_C(S)$ that we obtain if we allow only the constant b. In a similar way we define the other families of logical relations and boolean formulas that we obtain if we allow only the constant b.

We develop only the case $b = 0$. We show that in this case, if conditions (a) and (c) fail then $\exists\forall$-Formula$_0(S)$ is not 0-valid. The proof is divided in three parts:

First, we show that if a logical relation is not equality antimonotone, then it looses its 0-validity under quantification unless it can generate the logical relation $[x \oplus y \oplus z = 0]$. To start, we prove the same result for the non-horn logical relations that is included as a case in the full proof.

Lemma 12 *Let R be a non-horn logical relation of rank k. Then $[x \oplus y \oplus z = 0] \in \exists\forall$-Relation$_0(\{R\})$ or $\exists\forall$-Relation$_0(\{R\})$ is not 0-valid.*

Proof.
Let $T \subseteq \{1,\ldots,k\}$ be a subset of indices and t be a $|T|$-compatible assignment not in $R|T$ that contains at least two zeros (The existence of such assignment is guaranteed by Lemma 7). Let i and j be the indices of such zeros and V' be the set of the rest of variables with value zero. That is $V' = \{x_l | l \in T, l \neq i, l \neq j, t[l] = 0\}$. Let $F \in \exists$-Formula$(\{R\})$ be a boolean formula in the variables $\{x_1,\ldots,x_{|T|}\}$ equivalent to $R|T$. Let l be an index different from i and j. We construct the new formula $G = F\left[\begin{smallmatrix}V' \\ 0\end{smallmatrix}, \begin{smallmatrix}\{x_1,\ldots,x_{|T|}\}-V' \\ x_l\end{smallmatrix}\right]$. G is a boolean formula in the set of variables $\{x_i, x_j, x_l\}$. We assume that G is 0-valid, otherwise we have a logical relation in $\exists\forall$-Relation$_0(\{R\})$ that is not 0-valid. The truth table of $[G]$ does not contain $(0,0,1)$ but contains $(0,0,0)$ (by the 0-validity), $(1,0,1)$, and $(0,1,1)$ (by the $|T|$-compatibility, because this two assignments only change

the value of a variable in F) . We can derive that $(1,0,0) \notin [G]$, otherwise $(1,0,0) \leq (1,0,1)$ satisfy hypothesis of Lemma 10 and $[G]$ is not 0-valid. Analogously we can infer that $(0,1,0) \notin [G]$. $[G]$ is not complementive, therefore $(1,1,1) \notin [G]$ (otherwise we can apply Corollary 11). There are two possibilities according to whether $(1,1,0)$ belongs or not to $[G]$. In the first case $[G] = [x_i \oplus x_j \oplus x_l = 0]$, otherwise $\forall x_j \exists x_i G \equiv [x_l]$ and therefore $\exists\forall$-Relation$_0(\{R\})$ is not 0-valid. ■

Right now, we state the claimed result for the equality antimonotone logical relations:

Lemma 13 *Let R be a non-equality antimonotone logical relation. Then $[x \oplus y \oplus z = 0] \in \exists\forall$-Relation$_0(\{R\})$ or $\exists\forall$-Relation$_0(\{R\})$ is not 0-valid.*

Proof.
Let be G a boolean formula that expresses relation R. Repeat step A while it is applicable.

[Step A] If there exists indices i and j such that for all $t \in [G]$, $t[i] = t[j]$. Assign $G := G\left[{\{x_i,x_j\} \atop x_i}\right]$.

By Lemma 9 G is not antimonotone. Let T be a subset and t an assignment such that condition (c) in Lemma 8 fails. Let F be a boolean formula over the variables $\{x_1, \ldots, x_{|T|}\}$ equivalent to $[G]|T$. We can suppose that t contains only one zero, otherwise F is not horn and we can apply directly Lemma 12. We study three cases:

- $|T| = 1$. In this case $t = 0$, $F \equiv [x]$ and therefore $\exists\forall$-Relation$(\{R\})$ is not 0-valid.

- $|T| = 2$. Suppose that $t = (1,0)$. In this case $(0,0),(1,1) \in [F]$ (by the $|T|$-compatibility) and $(1,0) \notin [F]$. We know that $(0,1) \in [F]$, otherwise Step A is applicable. Let x be the variable in t with value 1, then $\forall x F$ is not 0-valid.

- $|T| \geq 3$. In this case, let i be the index corresponding to the value 0 in t. By the $|T|$-compatibility we know that $t_{0 \leftarrow j} \in [F]$ when $1 \leq j \neq i \leq |T|$. Therefore for every $t' \neq t$ such that $t'[i] = 0$, $t' \in [F]$, otherwise F is not horn by Lemma 7 since $t' = \bigwedge_{j \neq i, t'[j]=0} t_{0 \leftarrow j}$ and we can apply Lemma 12. By the $|T|$-compatibility $1^{|T|} \in [F]$. Thus, F is complementive (otherwise $[F]$ is not 0-valid by Corollary 11) and F does not contain exactly the two assignments t and $\overline{t}$. To show that

$\exists\forall$-Relation$(\{[F]\})$ is not 0-valid we only need to consider for some $1 \leq j \neq i \leq k$ the assignments $0^{|T|}_{1\leftarrow j} \leq 0^{|T|}_{1\leftarrow j, 1\leftarrow i}$ and apply directly Lemma 10.

■

Second, we show that if a logical relation is not 0-affine, then it looses its 0-validity under quantification unless it can generate the logical relation $[\overline{x} \vee \overline{y}]$.

Lemma 14 *Let R be a non- 0-affine logical relation of rank k. Then $[\overline{x} \vee \overline{y}] \in \exists\forall$-Relation$_0(\{R\})$ or $\exists\forall$-Relation$_0(\{R\})$ is not 0-valid.*

Proof.
If condition (bi) in Lemma 6 fails, R is not 0-valid. Otherwise, let t_1 and t_2 be assignments such that condition (bii) in Lemma 6 fails. Let $F \in \exists$-Relation$_0(\{R\})$ be a boolean formula defined in the following way:

$$F(x, y, z) = R(x_1, \ldots, x_k)\begin{bmatrix} V_{00} & V_{10} & V_{01} & V_{11} \\ 0 & x & y & z \end{bmatrix},$$

where $V_{ij} = \{x_l | t_1[l] = i \wedge t_2[l] = j\}$.

Note that $(0, 0, 0)$, $(1, 0, 1)$, and $(0, 1, 1)$ are in $[F]$ but not $(1, 1, 0)$. If $(1, 1, 1) \notin [F]$, then $\exists z F \equiv (\overline{x} \vee \overline{y})$. Otherwise, by Corollary 11, F must be complementive, $[F]$ contains $(0, 1, 0)$ and $(1, 0, 0)$ but not $(0, 0, 1)$. And then $\forall z F \equiv (x \not\equiv y)$ is not 0-valid. ■

Third, we prove that from $[x \oplus y \oplus z = 0]$ and $[\overline{x} \vee \overline{y}]$ we can generate a relation which is not 0-valid.

Lemma 15 *The set $\exists\forall$-Relation$(\{[\overline{x} \vee \overline{y}], [x \oplus y \oplus z = 0]\})$ is not 0-valid.*

Proof. Consider the formula $\forall y \exists x (x \oplus y \oplus z = 0) \wedge (\overline{x} \vee \overline{y}) \equiv z$. ■

The only thing that remains to do to complete the proof is to prove that if conditions (a) and (c) fail, then we can simulate the constant $b = 0$. We prove this in two stages.

Lemma 16 *Let R be a non 0-affine logical relation of rank k. Then the class $\exists\forall$-Relation$(\{R\})$ is not 0-valid or not 1-valid.*

Proof. By contradiction. Assume that the logical relation R is not 0-affine and $\exists\forall$-Relation$(\{R\})$ is both 0-valid and 1-valid. Let $t_1, t_2 \in R$ be assignments such that condition (bii) in Lemma 6 is falsified. If $t_1 \leq t_2$ or viceversa then by Lemma 10, $\exists\forall$-Relation$(\{R\})$ is not 0-valid. Otherwise, let T_1, T_2 nonempty disjoint sets of indices defined by $T_1 = \{i | t_1[i] = 1 \wedge t_2[i] = 0\}$ and $T_2 = \{i | t_1[i] = 0 \wedge t_2[i] = 1\}$. Note that $t_1 \oplus t_2 = 0^k_{1 \leftarrow T_1 \cup T_2}$.

We know that $t' = 0^k_{1 \leftarrow T_1} \in R$. Otherwise $\exists x R(x_1, \ldots, x_k) \left[{x_i : i \in T_2 \atop x} \right]$ contains $t'_1 = t_1 | (\{1, \ldots, k\} - T_2)$ and $t'_2 = t_2 | (\{1, \ldots, k\} - T_2)$ but not $t'_1 \oplus t'_2$. We know that $t'_2 \leq t'_1$ and therefore by Lemma 10, $\exists\forall$-Relation$(\{R\})$ is not 0-valid. In a similar way we can prove that $0^k_{1 \leftarrow T_2} \in R$. By the 1-validity of R, we know that $1^k \in R$. Therefore by Corollary 11 R is complementive and $t'' = 1^k_{0 \leftarrow T_2} \in R$ and $t''' = 1^k_{0 \leftarrow T_1 \cup T_2} \notin R$. Thus, we have two assignments $t' \leq t'' \in R$ such that $t''' = t' \oplus t'' \notin R$. By Lemma 10, $\exists\forall$-Relation$(\{R\})$ is not 0-valid. ■

Corollary 17 *Let R be a logical relation. If R is not 0-affine and not equality antimonotone then $\exists\forall$-Relation$(\{R\})$ is not 0-valid.*

Proof. By contradiction. Assume that $\exists\forall$-Relation$(\{R\})$ is 0-valid, by Lemma 16, $\exists\forall$-Relation$(\{R\})$ is not 1-valid. Therefore, we can generate the logical relation $\overline{x} \equiv R_1(x, \ldots, x)$ taking some non-1-valid logical relation $R_1 \in \exists\forall$-Relation$(\{R\})$. With $[\overline{x}]$ we can simulate the constant 0 in every formula by the construction $\exists x_0 (F \wedge \overline{x_0}) \left[{0 \atop x_0} \right]$ and therefore we have $\exists\forall$-Relation$(\{R\})$ = $\exists\forall$-Relation$_0(\{R\})$. But $\exists\forall$-Relation$_0(\{R\})$ contains $[\overline{x} \vee \overline{y}]$ (by Lemma 14) and $[x \oplus y \oplus z = 0]$ (by Lemma 13) and therefore is not 0-valid (by Lemma 15). ■

The dual result is stated without proof.

Corollary 18 *Let R be a logical relation. If R is not 1-affine and not equality monotone then $\exists\forall$-Relation$(\{R\})$ is not 1-valid.*

Finally, Theorem 4 follows from Corollaries 17 and 18.

6.2 Proof of Theorem 5

Learning Preliminaries

Most of the terminology about learning comes from [2]. Binary strings will represent both examples and concept names. If x is a string and $b \in \{0, 1\}$ is a constant, $|x|$ denotes its length and $|x|_b$ 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* $\mathcal{C}$ 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 equivalences queries [1] and the model of PAC-prediction with membership queries as defined by Angluin and Kharitonov in [2].

To compare the difficulty of learning problems in the prediction model we use the *prediction-preserving reducibility with membership queries* as defined by Angluin and Kharitonov [2]. It is denoted by $\leq_{pwm}$ and it extends Pitt and Warmuth's *prediction-preserving reducibility* [19] to the presence of membership queries.

Definition 19 *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 [2].

The only property of this reducibility that is needed in this paper was shown in [2]:

Lemma 20 *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.*

In the proof of Theorem 5 we use a previous dichotomy theorem for learning quantified boolean formulas with constants that we proved in [8].

Theorem 21 *[8](Dichotomy Theorem for the learnability of quantified formulas with constants) Let S a finite set of logical relations. If S satisfies one of the conditions (a)-(d) below, then $\mathcal{C}_{\exists\forall\text{-}Formula_C(S)}$ is polynomially exactly learnable with improper equivalence queries. Otherwise, $\mathcal{C}_{\exists\text{-}Formula_C(S)}$ 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.

Proof of Theorem 5

The learnable cases follow directly from Theorem 21, since the problem without constants becomes easier and therefore we can use the algorithm for learning formulas with constants to learn formulas without constants.

The non-learnability results follow from Theorem 4 taking into account that the equality antimonotone (resp. equality monotone) relations are a subset of weakly antimonotone (resp. weakly monotone relations) and that the 0-affine and 1-affine relations are a subset of affine relations. Therefore, if conditions $(a) - (d)$ in Theorem 21 do not hold, then conditions (e) or (f) in Theorem 4 hold. In both cases we can prove the following pwm-reduction

$$\mathcal{C}_{\exists\forall\text{-Formula}_C(S)} \leq_{pwm} \mathcal{C}_{\exists\forall\text{-Formula}(S)}$$

Then by Lemma 20, the negative results follow.

For every natural number n and s and for every concept name $u \in \mathcal{C}_{\exists\forall\text{-Formula}_C(S)}$ such that $|u| \leq s$, let $F \in \exists\forall\text{-Formula}_C(S)$ be the quantified boolean formula represented by u.

If condition (e) holds then we can generate the relations $[x]$ and $[\overline{x}]$. Let u' be the representation of the boolean formula formed from $G = \exists x_0 \exists x_1 (F \wedge \overline{x_0} \wedge x_1) \left[\begin{smallmatrix} 0 \\ x_0 \end{smallmatrix}, \begin{smallmatrix} 1 \\ x_1 \end{smallmatrix}\right]$ where x_0 and x_1 are variables not appearing in F, so $F \equiv G$. We define $g(s, n, u) = u'$, $f(s, n, x) = x$, and $h(s, n, x) = x'$. Clearly, f, g, and h satisfy conditions (1), (2) and (3) in Definition 19 and therefore define a pwm-reduction.

Otherwise, condition (f) holds. Therefore every boolean formula in $\exists\forall\text{-Formula}(S)$ is complementive and we can generate the relation $[x \not\equiv y]$. Let u' the representation of the boolean formula formed from $G = F \wedge (x_0 \not\equiv x_1) \left[\begin{smallmatrix} 0 \\ x_0 \end{smallmatrix}, \begin{smallmatrix} 1 \\ x_1 \end{smallmatrix}\right]$ where x_0 and x_1 are variables not appearing in F and without loss of generality we assume that x_0 and x_1 are, in this order, the last two variables in lexicographical order of G. We define $g(s, n, u) = u'$. We define $f(s, n, x) = x'$ where $x' \in X$ is the assignment constructed by adding 01 at the end of x. Let x' be the assignment obtained by removing the last two components of x if possible. We define $h(s, n, x) = x'$ if $|x| \geq 2$ and the last two components of x are 01, $\overline{x'}$ if $|x| \geq 2$ and the last two components of x are 10, and $\perp$ otherwise. Clearly, f, g, and h satisfy conditions (1), (2) and (3) in Definition 19 and therefore define a pwm-reduction. ■