Learnability of Quantified Formulas

Víctor Dalmau *
Informática e Ingenieria de Sistemas, Universidad de Zaragoza
Centro Politécnico Superior
Maria de Luna 3
50015 Zaragoza, Spain
dalmau@lsi.upc.es

Peter Jeavons
Computing Laboratory, Oxford University
Wolfson Building, Parks Road
Oxford, OX1, 3QD, England
Peter.Jeavons@comlab.ox.ac.uk

Abstract

We consider the following classes of quantified formulas. Fix a set of basic relations called a basis. Take conjunctions of these basic relations applied to variables and constants in arbitrary ways. Finally, quantify existentially or universally some of the variables. We introduce some conditions on the basis that guarantee efficient learnability. Furthermore, we show that with certain restrictions on the basis the classification is complete. We introduce, as an intermediate tool, a link between this class of quantified formulas and some well-studied structures in Universal Algebra called clones. More precisely, we prove that the computational complexity of the learnability of these formulas is completely determined by a simple algebraic property of the basis of relations: their clone of polymorphisms. Finally, we use this technique to give a simpler proof of the already known dichotomy theorem over Boolean domains.

1 Introduction

The problem of learning an unknown formula under some determined protocol has been widely studied. The inevitable trade-off between the

*This work has been partly supported by: the Spanish Government through the DGES under project PB95-0787 (KOALA), the U.K. Government through the EPSRC under grant GR/L099636, the Catalan Government through CIRIT under the project 1997SGR-00366 (SGR), and the European Commission through the ESPRIT BRA Program under the following projects: Working Group 27150 (NeuroCOLT2), project 20244 (ALCOM IT) and project TIC97-1475-CE (CICYT).

expressive power of a family of formulas and the resources needed to learn them has forced researchers to study restricted classes of formulas. Among them, propositional formulas have received particular attention. It is known that learning general propositional formulas is hard [3, 18] in the usual learning models, but some efficiently learnable subclasses of Boolean formulas, especially inside CNF and DNF, have been identified (see [1, 2, 5, 13] for example).

First-order logic is a formalism with superior expressive power, but it is not so well studied from the computational learning point of view (see [19] and further references in that paper). An active line of research in predicate logics is, for instance, *Inductive Logic Programming (ILP)* [21, 22].

In a recent paper, Dalmau [9] introduced a framework to study the learnability of quantified Boolean formulas and proved a complete classification for finite bases. (This result was inspired by a well known dichotomy on the satisfiability of Boolean formulas proved by Schaefer [28].) The main goal of this paper is to further continue that line of research by extending those results to domains of arbitrary size, since quantification makes more sense when it can be applied to arbitrary variables, rather than Boolean variables only.

As a main intermediate tool for our study we introduce a link with some well known algebraic structures, called *clones* in Universal Algebra. This approach has been introduced in a different context by Jeavons et al. [16] and it has been successful in the study of the Constraint Satisfaction Problem (CSP) [14, 15, 16, 17]. We prove that the learning complexity of a family of quantified formulas over a finite domain is completely determined by its *clone of polymorphisms*.

As a first application of this new technique, we introduce two families of efficiently learnable classes, namely, coset generating (CG) and near-unanimity (NU) bases. These contain, as a particular case, the learnable classes for the Boolean domain. Furthermore, we show that if we restrict the formulas in certain ways, this classification is complete.

Despite the fact that there exists a dichotomic classification for the Boolean domain, a full dichotomy for arbitrary domains is not known and seems likely to be hard to find, since the clone lattice that characterizes the learnability of quantified formulas is rather involved; actually it is uncountable in the general case of a basis with a possibly infinite number of relations. For the Boolean domain, the clone lattice is simpler and it has been completely characterized [29, 26].

The positive learnability results are obtained using a very simple algorithm, which we call the *Generating Set (GS) Algorithm*. This algorithm exploits the intersection closure property of some representation classes. Learnable classes are shown to be efficiently learnable with equivalence queries in the model of exact learning with queries. This feature, which also appears in the Boolean case, is rather striking, since in all our dichotomic classifications the rest of the classes are shown not to be polynomially learnable even in the more powerful model of PAC-prediction with membership queries.

Another characteristic feature of the learnable classes is that every concept in them can be described as the minimum concept containing a set of examples with size polynomial in the number of attributes (variables). This fact has some interesting consequences. First, the computational complexity of learning these classes only depends on the number of attributes but not on the length of the particular representation, strengthening the dichotomy. Second, the total number of concepts with a determined number of attributes is a number singly exponential in the number of attributes.

The structure of the paper is as follows. In Section 2 we define the framework of quantified formulas that we are using throughout, and in Section 3 we give the necessary background in computational learning theory. In Section 4 we introduce the GS algorithm, and give sufficient conditions to ensure that this algorithm works efficiently. In Section 5 we identify a powerful link with universal algebra, which is the main novel feature of this paper, and in Section 6 we use algebraic results to identify two families of efficiently learnable classes. In Section 7 we use further algebraic results to establish some non-learnability results, and obtain complete classifications for the learnability of quantified formulas in certain restricted cases. Finally, In Section 8, we demonstrate how the algebraic results developed in this paper provide a much simpler proof for the classification of the learnability of quantified Boolean formulas.

2 Generalized Quantified Formulas

Let $V = \{x_1, x_2, \dots\}$ be an infinite set of variables. A *literal* is a variable or its negation. Let D be a finite set called the *domain*. For convenience, we will assume that the domain has the form $D = \{0, 1, \dots, |D|-1\}$, since any result over $\{0, 1, \dots, |D|-1\}$ can be moved straightforwardly to any arbitrary domain with the same size.

Let D^* be the set of all vectors over D. An *assignment* or *string* is a vector in D^*. If t is an assignment over D , $|t|$ denotes its length. For any assignment $t \in D^*$ and for any integer $1 \leq j \leq |t|$, $t[j] \in D$ denotes the jth component of t. For any natural number k, $D^k = \{t \in D^* : |t| = k\}$. A k-ary *relation* (k integer) over D is a subset of D^k where k is called the *rank* or *arity* of the relation. The set $\{1, \dots, k\}$ is referred to as the set of *indices* or *coordinates* of R. Relation eq_D is defined by:

$$\mathrm{eq}_D = \{\langle x, x\rangle : x \in D\}$$

The domain containing exactly two elements, i.e., $\{0, 1\}$, is called the Boolean domain and denoted $\mathcal{B}$. Value 0 corresponds to *false* and value 1 corresponds to *true*. Relations over the Boolean domain are called *Boolean* or *logical* relations. Strings (assignments) over Boolean domains are called *binary* strings (assignments).

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

Definition 1 Let D be any finite set and let $\Gamma = \{R_1, R_2, \dots\}$ be any set of relations over D, where each R_i has rank k_i. (R_i denotes both the relation and its symbol.)

The set of *generalized quantified formulas with constants over the basis* Γ, denoted by $\exists\forall\text{-}\operatorname{Form}_C(\Gamma)$, is the smallest set of formulas such that:

a.- For all $R \in \Gamma$ of rank k, $R(y_1, \ldots, y_k) \in \exists\forall\text{-}\operatorname{Form}_C(\Gamma)$ where $y_i \in V \cup D$ for $1 \leq i \leq k$.

b.- For all $\Phi, \Psi \in \exists\forall\text{-}\operatorname{Form}_C(\Gamma)$, $\Phi \wedge \Psi \in \exists\forall\text{-}\operatorname{Form}_C(\Gamma)$.

c.- For all $\Phi \in \exists\forall\text{-}\operatorname{Form}_C(\Gamma)$ and for all $x \in V$, $\exists x \Phi \in \exists\forall\text{-}\operatorname{Form}_C(\Gamma)$.

d.- For all $\Phi \in \exists\forall\text{-}\operatorname{Form}_C(\Gamma)$ and for all $x \in V$, $\forall x \Phi \in \exists\forall\text{-}\operatorname{Form}_C(\Gamma)$.

If we remove condition **(d)** in the previous definition we obtain a reduced class of formulas called *generalized existentially quantified formulas with constants over the basis* Γ, denoted by $\exists\text{-}\operatorname{Form}_C(\Gamma)$. Furthermore, if we also remove condition **(c)** we obtain a more reduced class called *generalized formulas with constants over the basis* Γ and denoted by $\operatorname{Form}_C(\Gamma)$.

If in the previous definitions we replace "$y_i \in V \cup D$" by "$y_i \in V$" in **(a)**, we obtain the constant-free counterpart of the previous sets of formulas denoted by $\exists\forall\text{-}\operatorname{Form}(\Gamma)$, $\exists\text{-}\operatorname{Form}(\Gamma)$, $\operatorname{Form}(\Gamma)$ respectively.

Each formula Φ defines a relation $[\Phi]$ if we apply the usual semantics of first-order logic, and the variables are taken in lexicographical order. More formally, let Φ be a formula over the free variables $x_{i_1}, x_{i_2}, \ldots, x_{i_m}$ where $i_1 < i_2 < \cdots < i_m$, then

$$[\Phi] = \{\langle d_1, d_2, \ldots, d_m \rangle : x_{i_j} = d_j \ (1 \leq j \leq m) \text{ satisfies } \Phi\}$$

This operator can be extended to sets of formulas: for every set of relations Γ we define $\operatorname{Rel}(\Gamma) = \{[\Phi] : \Phi \in \operatorname{Form}(\Gamma)\}$. Similarly, we also define $\exists\forall\text{-}\operatorname{Rel}(\Gamma)$, $\exists\text{-}\operatorname{Rel}(\Gamma)$, $\operatorname{Rel}_C(\Gamma)$, $\exists\forall\text{-}\operatorname{Rel}_C(\Gamma)$, and $\exists\text{-}\operatorname{Rel}_C(\Gamma)$.

3 Learning Preliminaries

Most of the terminology used in this section comes from Angluin and Kharitonov [3]. Let D be a fixed domain and let $X = D^*$ be the set of strings over D. Strings in X will represent both examples and concept names.

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

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

For any set of formulas $\mathcal{F}$, we define $\mathcal{C}_{\mathcal{F}}$ to be 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.

We use two models of learning, both of them fairly standard: Angluin's model of exact learning with queries defined by Angluin [1] and the model of PAC-prediction with membership queries as defined by Angluin and Kharitonov [3].

3.1 The Exact Query Learning Model (EQ)

Let $\mathcal{H}$ be a representation class. An *exact learning algorithm with queries* is an algorithm A that takes as input a bound s on the size of the target concept representation. 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 *membership* and *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 1 if $c = K_{\mathcal{H}}(h)$ and a counterexample $x \in c \triangle K_{\mathcal{H}}(h)$ otherwise.

The algorithm A *runs in polynomial time* if its running time (counting one step for each oracle call) is bounded by a polynomial in s and the length of the largest counterexample. We say that A *exactly learns* a representation of concepts $\mathcal{C}$, if and only if for all positive integers s, for all concept names u with $|u| \leq s$, when A is run with input s, and oracles determined by $c = K_{\mathcal{C}}(u)$, A outputs a concept name v such that $c = K_{\mathcal{H}}(v)$. If $\mathcal{C} = \mathcal{H}$ then we say that A *properly* learns $\mathcal{C}$, otherwise we say that A *improperly* learns $\mathcal{C}$.

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 exactly learns $\mathcal{C}$.

3.2 The PAC-prediction model

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 an unknown probability distribution $\mathcal{D}$ on D^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 $\mathcal{D}$ 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 $\mathcal{D}$.

The algorithm 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. We say that 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 *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 $u \in X$ with $|u| \leq s$, when A is run with inputs s, n, and ϵ, and oracles determined by $c = K_{\mathcal{C}}(u)$ and $\mathcal{D}$, 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 the PAC sense, with the additional help of membership queries.

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 predicts $\mathcal{C}$.

3.2.1 Reducibility among prediction problems

To compare the difficulty of learning problems in the prediction model we use *prediction-preserving reducibility with membership queries* as defined by Angluin and Kharitonov [3]. It is denoted by $\leq_{pwm}$ and it extends Pitt and Warmuth's [25] *prediction-preserving reducibility* to include 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 $\mathcal{C} \leq_{\text{pwm}} \mathcal{C}'$, 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 $u \in X$ with $|u| \leq s$, $g(s, n, u)$ is a string u' of length at most $q(s, n, |u|)$.
2. For all natural numbers s and n, for every string $u \in X$ with $|u| \leq s$, and for every x with $|x| \leq n$, $f(s, n, x)$ 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 $u \in X$ with $|u| \leq s$, 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 [3].

The only properties of this reducibility that are needed in this paper were established in [3]:

Lemma 1 *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 2 *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.*

4 Generating Set Algorithm

In order to derive learning algorithms for some subclasses of quantified formulas we identify some special features of quantified formulas and we consider the more general problem of learning any representation class with those features. Finally we get results about learnability of quantified formulas as a particular case.

The first important feature that we use is given by the fact that every class of generalized formulas is closed under the Boolean operator $\wedge$, or equivalently, the class of concepts, as sets of examples, represented by quantified formulas, is closed under intersection. Intersection-closed classes of concepts have been intensively studied. In this section we discuss briefly intersection-closed classes in general. More precisely, we introduce an algorithm for learning intersection-closed representation classes with equivalence queries, and we present a simple condition that guarantees that a representation class is polynomially learnable using this algorithm.

We start by giving a formal definition of intersection-closed classes.

Definition 3 We say that a representation class $\mathcal{C}$ is intersection-closed if:

$$\forall c_1, c_2 \in K_{\mathcal{C}} \Rightarrow \exists c_3 \in K_{\mathcal{C}} \quad K_{\mathcal{C}}(c_3) = K_{\mathcal{C}}(c_1) \cap K_{\mathcal{C}}(c_2)$$

Let $\mathcal{C}$ be an intersection-closed representation class and let H be a set of examples. The intersection of all the concepts in $\mathcal{C}$ containing H, denoted by $\langle H \rangle_{\mathcal{C}}$ is the concept *generated* by H. We will say that H is a *generating set* of the concept $\langle H \rangle_{\mathcal{C}}$.

Notice that closure under intersection depends only on the set of concepts $K_{\mathcal{C}}$ and does not depend at all on the particular representation class $\mathcal{C}$. In fact, we can consider generating sets as an alternative representation class for the same collection of concepts. More formally, given a set of examples H, we say that an example x belongs to the class represented by H iff $x \in \langle H \rangle_{\mathcal{C}}$.

For many purposes a representation class has to be *polynomial-time evaluable*, that is, it has to be decidable in polynomial time whether a vector belongs to the class. For the remainder of this section we will assume that we are dealing with polynomial-time evaluable representation classes, and we will demonstrate later that this property holds for each concrete example we present.

For any intersection-closed representation class there is an obvious learning algorithm using equivalence queries: start with an empty set of generators and keep asking equivalence queries and adding vectors until the set is complete. We call this algorithm the *Generating Set (GS) Algorithm* and we state it in Figure 4 for future reference.

Algorithm **GS** can be applied to any intersection-closed representation class $\mathcal{C}$. There is a canonical algorithm for learning intersection-closed classes in the PAC model [31] called the *Closure Algorithm*: the output of this algorithm is always $\langle H \rangle_{\mathcal{C}}$ where H contains all the positive examples [23]. Algorithm **GS** is a direct adaptation of the Closure Algorithm.

Algorithm **GS**
$H := \emptyset$,
while $EQ(\langle H\rangle_{\mathcal{C}})=$ 'no' do
    Let c be the counterexample
    $H := H \cup \{c\}$
endwhile
return $\langle H\rangle_{\mathcal{C}}$

Figure 1: Generating Set Algorithm (**GS**)

Consequently, it is possible to extend the results in this paper to nested differences of intersection-closed classes using an approach similar to [11].

It is possible to convert algorithm **GS** to a proper algorithm finding, for every set of generators H, a concept c equivalent to $\langle H\rangle_{\mathcal{C}}$, but this step can be computationally expensive.

Clearly, algorithm **GS** always finds the target concept. The only drawback is its time complexity, since it can require an exponential number of equivalence queries. In consequence, we are interested in characterizing the cases in which algorithm **GS** learns a representation class with a polynomial number of queries.

Notice that, when formulas are used as a representation class, all the examples of a given concept have the same size. A representation class is said to be *stratified* if it satisfies this condition. From now on, we will restrict to stratified representation classes.

We need the following definition:

Definition 4 Let $\mathcal{C}$ be a stratified intersection-closed representation class of concepts, and let $u = (x_0, x_1, \ldots, x_m)$ be a list of tuples in D^n (where D is the alphabet of the examples). If for every $1 \leq i \leq m$, $x_i \notin \langle\{x_0, \ldots, x_{i-1}\}\rangle_{\mathcal{C}}$ then we call u an *additive sequence* over $\mathcal{C}^n$.

Clearly, an additive sequence is just a different reformulation of a possible sequence of counterexamples provided by algorithm **GS**. A representation class is learnable using algorithm **GS** if the size of every possible sequence of counterexamples can be bounded by a polynomial in the size of the examples and the size of the representation of the target concept. We do not consider the size of the representation of the target concept. This choice makes the analysis simpler, since we obtain independence from the particular representation class. Furthermore, as we will see later (Section 7), there is some evidence that this restriction does not make any difference when we are dealing with quantified formulas. Formally, we say that an stratified intersection-closed representation class $\mathcal{C}$ is *polynomially bounded* if every additive sequence over $\mathcal{C}^n$ has size polynomial in n.

Theorem 1 *Let $\mathcal{C}$ be a polynomially bounded representation class. Then $\mathcal{C}$ is polynomially learnable with improper equivalence queries using algorithm* **GS**. *Furthermore, $\mathcal{C}$ is also learnable with a polynomial number of proper equivalence queries (not necessarily in polynomial time).*

Finally, notice that every intersection-closed representation class contained in a polynomially bounded representation class is also polynomially bounded.

5 Learning Subuniverses

In this section we will refine the approach started in the previous section by studying a particular case of a stratified intersection-closed representation class: the set of subuniverses of the direct powers of some algebra. This new representation class is closely related to quantified formulas in a way that will be made precise later. First we introduce some algebraic notation.

5.1 Some Algebraic Preliminaries

The following definitions are fairly standard (see [20] for example).

Definition 5 An *algebra* $\mathbf{A}$ is an ordered pair $(A; F)$ such that A is a nonempty set, called the *universe*, and $F = \langle f_i : i \in I \rangle$ where f_i is a finitary operation on A for each $i \in I$. A is called the *universe* of $\mathbf{A}$, f_i is referred to as a *fundamental* or *basic operation* of $\mathbf{A}$ for each $i \in I$, and I is called the *index set* or the *set of operations symbols* of $(A; F)$.

Endowing algebras with indexed systems of operations rather than a mere set of operations has many advantages. Nevertheless, sometimes, it is most acceptable to display algebras following the customary viewpoint. For example, $(A; \cdot, 1, ^{-1})$ is a more comfortable way to define a group than $(A; \langle f_i : i \in \{\cdot, 1, ^{-1}\}\rangle)$.

If ϱ is an operation symbol of $\mathbf{A}$, then we use $\varrho^{\mathbf{A}}$ to stand for the fundamental operation of $\mathbf{A}$ indexed by ϱ. We say that ϱ *denotes* $\varrho^{\mathbf{A}}$ or that $\varrho^{\mathbf{A}}$ is the *interpretation* of ϱ on $\mathbf{A}$

Given an algebra $\mathbf{A}$ with index set I, there is a function Rank called the *rank function* from I to the set of natural numbers defined by:

$$\mathrm{Rank}(\varrho) \text{ is the rank of } \varrho^{\mathbf{A}} \text{ for all } \varrho \in I$$

The rank function of an algebra is also referred to as its *similarity type* or, more briefly, its *type*. Algebras $\mathbf{A}$ and $\mathbf{B}$ are said to be *similar* if and only if they have the same rank function.

There are some standard ways to assemble new algebras from those already at hand. The chief tools we will use are the formation of subuniverses and the formation of direct powers.

Let f be an m-ary operation over A, and let B be a subset of A. We say that B is closed under f (or f preserves B) if and only if

$$\forall a_1, \ldots, a_m \in B, \quad f(a_1, \ldots, a_m) \in B.$$

Definition 6 Let $\mathbf{A} = (A; F)$ be an algebra. A subset B of A which is closed under every operation in F is called a *subuniverse* of $(A; F)$. The algebra $\mathbf{B}$ is said to be a *subalgebra* of $\mathbf{A}$ if and only if $\mathbf{A}$ and $\mathbf{B}$ are similar, the universe B of $\mathbf{B}$ is a subuniverse of $\mathbf{A}$, and $\varrho^{\mathbf{B}}$ is the restriction to B of $\varrho^{\mathbf{A}}$, for each operation symbol ϱ of $\mathbf{A}$.

Note that the intersection of any two subuniverses of an algebra $\mathbf{A}$ is a subuniverse of the algebra $\mathbf{A}$.

Let f be a k-ary operation on A and let n be a positive integer. Then operation f^n is a k-ary operation on A^n defined by:

$$f^n(x_1, \dots, x_k) = \langle f(x_1[1], \dots, x_k[1]), \dots, f(x_1[n], \dots, x_k[n]) \rangle$$

Let $\mathbf{A} = (A; F)$ be an algebra and n any positive integer. The *direct power* of $(A; F)$ is the algebra, denoted by $\mathbf{A}^n$, with the same similarity type, with universe A^n such that for every operation symbol ϱ of $\mathbf{A}$, $\varrho^{(\mathbf{A}^n)} = (\varrho^{\mathbf{A}})^n$

Direct power is the natural way to extend an algebra to tuples of elements. From now on, due to the correspondence between an algebra and its direct powers we will do some notation abuse and for instance we will say that an n-ary relation R over A is closed under a k-ary operation $f : A^k \to A$, meaning that R is closed under f^n. This concept deserves to be stated as a definition.

Definition 7 Let f be an m-adic operation over A and let R be an n-ary relation over A. We say that f *preserves* R (or R *is closed under* f, or f *is a polymorphism of* R) if

$$t_1, t_2, \dots, t_m \in R \implies f^n(t_1, t_2, \dots, t_m) \in R.$$

5.2 Subuniverses as Concepts

Let $\mathbf{D} = (D; F)$ be a finite algebra. Any subuniverse B of $\mathbf{D}^n$ $(n \geq 1)$ can be regarded as a concept. The concept represented by B contains as examples precisely the n-ary tuples in B. Furthermore, the intersection of two subuniverses of $\mathbf{D}^n$ is a subuniverse of $\mathbf{D}^n$ as well. Thus, the class of all the subuniverses of $\mathbf{D}^n$ for *all* integers $n \geq 1$ is a stratified intersection-closed concept class. In consequence, the concepts introduced in the previous section, and in particular algorithm **GS**, are applicable to the subuniverses of direct powers of any finite algebra.

We define generating sets for subuniverses in an analogous way to the previous section. More precisely, let H be a subset of D^n, the intersection $\langle H \rangle_{(D;F)}$ of all subuniverses of $\mathbf{D}^n$ containing H will be called the *subuniverse generated* by H. Consequently, a reasonable way to represent subuniverses is by using generating sets.

Definition 8 We denote $\mathcal{C}_{(D;F)}$ the representation class given by the subuniverses of all direct powers $(D; F)^n$ $(n \geq 1)$ represented by sets of generators.

We say that an algebra $(D; F)$ is polynomially learnable (resp. polynomially evaluable, polynomially bounded) if so is $\mathcal{C}_{(D;F)}$. We say that an operation f on D is polynomially learnable (resp. polynomially evaluable, polynomially bounded) if so is $\mathcal{C}_{(D;f)}$.

Let us study, now, how subuniverses of direct powers are related to quantified formulas. Notice first, that the notion of polymorphism can be extended in a natural way to formulas. Consider for every quantified formula Φ over domain D with n free variables, its associated set of examples $[\Phi]$. By definition $[\Phi] \subseteq D^n$ is an n-ary relation over D. Thus, we say that a function f preserves Φ iff f preserves $[\Phi]$.

The key observation is the following. Let Γ be any arbitrary basis on domain D, and let F be the set of all functions preserving Γ. Then, under some technical assumptions that will be made precise later, every function $f \in F$ preserves every formula in $\exists\forall\text{-}\,\mathrm{Form}_C(\Gamma)$ as well. In consequence, every relation associated with a formula in $\exists\forall\text{-}\,\mathrm{Form}_C(\Gamma)$ is a subuniverse of a direct power of the algebra $(D; F)$.

Before we can state this observation in a more precise form we need to introduce some terminology. An operation f on D is said to be *exhaustive* if its image is the whole of D. Operation f is said to be *idempotent* if it satisfies $f(x, x, \ldots, x) = x$ for all $x \in D$. Idempotent operations are clearly exhaustive. For any set F of operations, we define $Idem(F)$ to be the subset containing exactly the idempotent operations in F.

Lemma 3 *Let Γ be a set of logical relations closed under some operation f over D. Then, $\exists\text{-}\,\mathrm{Rel}(\Gamma)$ is also closed under f. Furthermore, if f is exhaustive then $\exists\forall\text{-}\,\mathrm{Rel}(\Gamma)$ is closed under f. Furthermore, if f is idempotent then $\exists\forall\text{-}\,\mathrm{Rel}_C(\Gamma)$ is closed under f.*

As a consequence of this result, it is possible to turn results about subuniverses into results about quantified formulas by considering closure operations. The next theorem summarizes the results. Notice that we are not assuming finiteness in the basis $\Gamma = [\mathcal{F}]$.

Theorem 2 *Let $\mathcal{F}$ be a class of formulas over a finite domain D and let F be the set of all functions on D preserving $[\mathcal{F}]$. If algebra $(D; F)$ is polynomially bounded then the following conditions hold:*

- *(a) $\mathcal{F}$, $\mathrm{Form}([\mathcal{F}])$, and $\exists\text{-}\,\mathrm{Form}([\mathcal{F}])$ are polynomially learnable with improper equivalence queries.*
- *(b) $\mathrm{Form}([\mathcal{F}])$ and $\exists\text{-}\,\mathrm{Form}([\mathcal{F}])$ are learnable with a polynomial number of proper equivalence queries (not necessarily in polynomial time).*
- *(c) If every operation in F is exhaustive, then conditions (a) and (b) are also satisfied by $\exists\forall\text{-}\,\mathrm{Form}([\mathcal{F}])$.*
- *(d) If every operation in F is idempotent, then conditions (a) and (b) are also satisfied by $\mathrm{Form}_C([\mathcal{F}])$, $\exists\text{-}\,\mathrm{Form}_C([\mathcal{F}])$, and $\exists\forall\text{-}\,\mathrm{Form}_C([\mathcal{F}])$.*

Proof. Condition (a) is a direct consequence of the definition of polynomially bounded algebra, Lemma 3, and the first part of Theorem 1. To show condition (b) just consider that both classes of concepts are intersection-closed and apply the second part of Theorem 1. Condition (c) follows from the fact that closure under an exhaustive operation is preserved by universal quantification. Finally, condition (d) is due to the fact that closure under an idempotent operation is preserved by constantification, and that idempotentence implies exhaustivity. ∎

6 Some Learnable Algebras

In this section we introduce two families of operations, namely, coset generating operations and near-unanimity operations, such that the class of subuniverses of algebras containing one of these operations is polynomially bounded. In consequence, by Theorem 2, every class of quantified formulas with a basis preserved by some function in one of these families is polynomially learnable using algorithm **GS**.

6.1 Coset Generating Operations

Definition 9 An operation $\varphi : A^3 \to A$, is called a *'coset generating (CG) operation'* if for all $x, y, z, u \in A$,

1. $\varphi(x, x, y) = \varphi(y, x, x) = y$
2. $\varphi(\varphi(x, y, z), z, u) = \varphi(x, y, u)$
3. $\varphi(u, z, \varphi(z, y, x)) = \varphi(u, y, x)$

It is easy to adapt the proof of Proposition 2.2 in [29] to show that the previous definition is equivalent to the existence of some group $(A; \cdot, ^{-1}, 1)$ such that $\varphi(x, y, z) = x \cdot y^{-1} \cdot z$.

Theorem 3 *Let $\varphi : A^3 \to A$ be an operation. The following two conditions are equivalent:*

a. Operation φ is coset generating.

b. There exists a group $(A; \cdot, ^{-1}, 1)$ such that $\varphi(x, y, z) = x \cdot y^{-1} \cdot z$.

Proof. $(i)[(b) \implies (a)]$**.** A direct application of the properties of a group yields that

$$\varphi(x, x, y) = x \cdot x^{-1} \cdot y = y = y \cdot x^{-1} \cdot x = \varphi(y, x, x)$$
$$\varphi(\varphi(x, y, z), z, u) = (x \cdot y^{-1} \cdot z) \cdot z^{-1} \cdot u = x \cdot y^{-1} \cdot u = \varphi(x, y, u)$$
$$\varphi(u, z, \varphi(z, y, x)) = u \cdot z^{-1} \cdot (z \cdot y^{-1} \cdot x) = u \cdot y^{-1} \cdot x = \varphi(u, y, x).$$

$(ii)[(a) \implies (b)]$**.** Choose an element $1 \in A$ arbitrarily, and define operations $\cdot$ and $^{-1}$ by

$$x \cdot y = \varphi(x, 1, y) \text{ and } x^{-1} = \varphi(1, x, 1)$$

Now, we have to verify that the previous definitions satisfy all the group properties. First, we start showing that 1 is a neutral element

$$x \cdot 1 = \varphi(x, 1, 1) = x = \varphi(1, 1, x) = 1 \cdot x.$$

Furthermore, we have

$$x \cdot y^{-1} = \varphi(x, 1, \varphi(1, y, 1)) = \varphi(x, y, 1),$$

$$y^{-1} \cdot x = \varphi(\varphi(1, y, 1), 1, x) = \varphi(1, y, x).$$

Thus, the correctness of $^{-1}$ comes from

$$x \cdot x^{-1} = \varphi(x, x, 1) = 1 = \varphi(1, x, x) = x^{-1} \cdot x,$$

whence

$$(x^{-1})^{-1} = \varphi(1, x^{-1}, 1) = \varphi(x \cdot x^{-1}, x^{-1}, 1) =$$
$$= \varphi(\varphi(x, 1, x^{-1}), x^{-1}, 1) = \varphi(x, 1, 1) = x$$

Furthermore,

$$(x \cdot y^{-1}) \cdot z = \varphi(\varphi(x, y, 1), 1, z) = \varphi(x, y, z),$$
$$x \cdot (y^{-1} \cdot z) = \varphi(x, 1, \varphi(1, y, z) = \varphi(x, y, z).$$

Thus,

$$(x \cdot (y^{-1})^{-1}) \cdot z = \varphi(x, y^{-1}, z) = x \cdot ((y^{-1})^{-1} \cdot z).$$

Since $(y^{-1})^{-1} = y$ it follows that $\cdot$ is associative. Hence $(A; \cdot, ^{-1}, 1)$ is a group and $\varphi(x, y, z) = x \cdot y^{-1} \cdot z$. ∎

If $(A; \cdot, ^{-1}, 1)$ is *Abelian*, we have a particular kind of CG operation called an affine operation. Affine operations have been intensively studied in universal algebra (see [29] for example). It is well known (see [15] for example) that for every affine operation over a finite set A of prime size, the subuniverses of its direct power are exactly the subsets of A^n that can be expressed as a system of counting functions modulo $|A|$. The learnability of these formulas has already been established in [6] using a similar strategy. In this section we generalize this result to all CG operations.

It is straightforward to see that any subset of A^n (n positive integer) closed under a coset generating operation φ is a right coset of a subgroup of the power group $(A; \cdot, ^{-1}, 1)^n$, where $(A; \cdot, ^{-1}, 1)$ is the group associated with φ.

Theorem 4 *Let B be any nonempty subset of A^n (n positive integer) and let $(A; \cdot, ^{-1}, 1)$ be a group. Then, the following two conditions are equivalent:*

a. B is closed under $\varphi(x, y, z) = x \cdot y^{-1} \cdot z$.

b. B is a right coset of a subgroup of the direct power group $(A; \cdot, ^{-1}, 1)^n$.

Proof Sketch.
(i)[$(a) \Longrightarrow (b)$] Let x any arbitrary tuple in B. Define $G = \{y \cdot x^{-1} : y \in B\}$. It is easy to prove that $B = G \cdot x$ and that G is a subgroup.
(i)[$(a) \Longrightarrow (b)$] Straightforward. ∎

Lemma 4 *Let φ be a CG operation over a finite domain D. Then (D, φ) is polynomially bounded.*

Proof. Let the list $u = (x_0, \ldots, x_m)$ be any additive sequence over $(D; \varphi)^n$. Then $\langle\{x_0, \ldots, x_m\}\rangle_{(D;\varphi)}$ is a right coset of a subgroup J of the power group $(D; \cdot, ^{-1}, 1)^n$, so we can take x_0 as a representative of the coset and consider the set $H = \{x_0 \cdot x_0^{-1}, x_1 \cdot x_0^{-1}, \ldots, x_m \cdot x_0^{-1}\}$ as a generating set for the subgroup J. From the fact that $\{x_0, \ldots, x_m\}$ is an additive sequence we have that H is independent in the sense that no element in H can be generated from the remainder. Therefore, the cardinality of the subgroup is at least $2^m \leq |D|^n$, which gives a polynomial bound for m. ∎

Lemma 5 *Let φ be a CG operation over a finite domain D. Then (D,φ) is polynomially evaluable.*

Proof.

The proof of the polynomial-time evaluability mimics the proof of Theorem 32 in [10].

Let $H = \{x_0, \ldots, x_m\}$ be any set of tuples over D^n. Let $(D; \cdot, ^{-1}, 1)$ be the group associated to φ. As it has been pointed out in the previous proof, $\langle H \rangle_{(D;\varphi)}$ is a right coset of a subgroup J of the power group $(D; \cdot, ^{-1}, 1)^n$. We can take x_0 as a representative of the coset and it is not hard to prove that $\{x_0 \cdot x_0^{-1}, x_1 \cdot x_0^{-1}, \ldots, x_m \cdot x_0^{-1}\}$ is a generating set of the subgroup J. So, the problem is reduced to the problem of deciding whether a tuple y_0 belongs to a group represented by a set of generators.

Consider the tower of subgroups G_i, $0 \leq i \leq n$ where G_i is the subgroup of J obtained fixing the first i components to 1. A right coset representation of this tower can be efficiently constructed using algorithm 7 in [12][1].

Now we will present a polynomial-time algorithm that, given a right coset representation for the tower of subgroups, decides whether $y_0 \in G_0$, proving the polynomial-time evaluability of CG operations.

Let $a_1, \ldots, a_r$ be the right coset representatives of G_0 in G_1. Let $T = \{a_i : 1 \leq i \leq r, a_i[1] = y_0[1]\}$ be the subset of the representatives coinciding with y_0 in the first component. It is clear that if $|T| = 0$ then $y_0 \notin G_0$. On the other hand, we have $|T| \leq 1$, otherwise, let a_i, a_j be two different elements in T, then $(a_j \cdot a_i^{-1})$ has a 1 as first component and, in consequence, belongs to G_1 (incidentally, this fact proves that the size of the coset representation is not too large).

Therefore, let a_i be the unique representative in T. Since, $y_0 \in G_0$ iff $y_0 \cdot a_i^{-1} \in G_1$, now we proceed with $y_1 = y_0 \cdot a_i^{-1}$ and G_1 as we did before for y_0 and G_0 and so on. If during some step j of this process we find that there does not exist any coset representative of G_j in G_{j+1} coinciding with y_j in the $j+1$ component then we know that the answer is *no*. Otherwise, after n repetitions the answer is *yes*. ■

By virtue of the connection between subuniverses and generalized formulas presented in the previous section we get, as a direct consequence of the previous results, the following theorem.

Theorem 5 *Let φ be a coset generating operation, and let $\mathcal{F}$ be a set of formulas closed under φ, then $\mathcal{C}_{(D;\varphi)}$ is polynomial-time evaluable and polynomially learnable with proper equivalence queries. Furthermore, the class $\exists\forall\text{-}\mathrm{Form}_C([\mathcal{F}])$ is polynomially learnable with equivalence queries in $\mathcal{C}_{(D;\varphi)}$.*

---

[1] It is convenient to notice that algorithm 7 requires a polynomial procedure for testing membership in G_i, a condition that in our case is not satisfied, since we are precisely interested in this procedure for G_0. However, this condition can be relaxed noticing that all the elements generated by algorithm 7 already belong to G_0. In this case, membership in G_i is straightforward: just check that the first i components are equal to 1.

6.2 Near-unanimity Operations

The class of near-unanimity operations has been well studied.

Definition 10 [4] An operation $\varphi : A^k \to A$, is called a *'near-unanimity (NU) operation'* if for all $x, y \in A$,

$$\varphi(x, y, y, \ldots, y) = \varphi(y, x, y, \ldots, y) = \cdots = \varphi(y, y, \ldots, y, x) = y.$$

A near-unanimity operation of rank 3 is also called *majority* operation.

Let R be an n-ary relation over D and let $I = \{i_1, \ldots, i_k\}$ where $i_1 < i_2 < \cdots < i_k$ be a set of indices chosen from $\{1, \ldots, n\}$. The projection $\mathrm{pr}_I(R)$ is defined to be the k-ary relation

$$\mathrm{pr}_I(R) = \{\langle t[i_1], \ldots, t[i_k]\rangle | t \in R\}$$

The projection of a tuple t, $\mathrm{pr}_I(t)$ is defined similarly.

Definition 11 An n-ary relation R over D is said to be r-decomposable if it contains all n-tuples t such that $\mathrm{pr}_I(t) \in \mathrm{pr}_I(R)$ for all set of indices I, from the set $\{1, 2, \ldots, n\}$ with $|I| \leq r$.

From [4] we have this useful property.

Theorem 6 *[4] Every relation R over D which is closed under a near-unanimity operation of arity r is $(r-1)$-decomposable.*

For near-unanimity operations it is even possible to prove proper learnability.

Theorem 7 *Let φ be a near-unanimity operation, and let $\mathcal{F}$ be a set of formulas closed under φ, then $\mathcal{C}_{(D;\varphi)}$ is polynomial-time evaluable and polynomially learnable with proper equivalence queries. Furthermore, the class $\exists\forall$-$\mathrm{Form}_C([\mathcal{F}])$ is polynomially learnable with proper equivalence queries.*

The proof of the theorem is split into several results.

Lemma 6 *Let φ be a NU operation over a finite domain D. Then $(D;\varphi)$ is polynomially bounded.*

Proof. Let r be the arity of φ and let $u = (x_0, \ldots, x_m)$ be any additive sequence over $(D;\varphi)^n$. For every $1 \leq i \leq m-1$, the set $H_i = \langle\{x_0, \ldots, x_i\}\rangle_{(D;\varphi)}$ is $(r-1)$-decomposable, by Theorem 6. We know that $x_{i+1} \notin H_i$, so there exists some set of indices I, with $|I| \leq r-1$ such that $\mathrm{pr}_I(x_{i+1}) \notin \mathrm{pr}_I(H_i)$. This implies that $\mathrm{pr}_I(x_{i+1}) \neq \mathrm{pr}_I(x_j)$ for every $0 \leq j \leq i$. The result follows from the fact that there are only a polynomial number of choices for $\mathrm{pr}_I(x)$. ■

The intuition underlying this result is the following: every relation closed under a near-unanimity operation of arity r is decomposable as a conjunction of relations of fixed arity $r-1$, therefore the problem of learning this class of relations is reduced to the problem of learning conjunctions of clauses of a fixed arity that can be solved using a similar approach to the one that is known for learning $(r-1)$-CNF [1]. An empty basis can be regarded as a conjunction containing a full $(r-1)$-ary relation for every possible set with at most $r-1$ indices. Every time we add a tuple to the basis, we remove in every relation the tuples falsified by the new tuple. We have a polynomial number of such tuples and with every addition to the basis we remove at least one.

Finally, we prove the polynomial-time evaluability of near-unanimity operations.

Lemma 7 *Let φ be a NU operation over a finite domain D. Then $(D;\varphi)$ is polynomially evaluable.*

Proof. Let r be the arity of φ, let H be a set of tuples over D^n and let x be a tuple over D^n. It is easy to see that $x \in \langle H\rangle_{(D;\varphi)}$ iff for every list of indices I over $\{1,2,\ldots,n\}$ with $|I| \leq r-1$, there exists some tuple u in H, such that $\mathrm{pr}_I(u) = \mathrm{pr}_I(x)$. Clearly, this condition can be checked in polynomial time. ∎

Finally we have to prove that it is possible to get proper learnability for quantified formulas. In fact, we establish a general condition that guarantees proper learnability in some more cases.

Lemma 8 *Let $\mathcal{F}$ be a class of formulas closed under a near-unanimity operation φ of arity r and closed under existential quantification, conjunction and renaming of variables. Then $\mathcal{F}$ is polynomially learnable with proper equivalence queries.*

Proof. We only need to prove that for every set H of tuples over D^n, it is possible to find a formula equivalent to $R = \langle H\rangle_{\mathcal{C}_\mathcal{F}}$ in polynomial time. Consider for every set $I = \{i_1,\ldots,i_{r-1}\}$ of $r-1$ indices, the relation $\mathrm{pr}_I\, R$. Clearly, this relation is obtained by some formula $\Phi_{\mathrm{pr}_I\, R}$ in $\mathcal{F}$ (the set of relations of rank $r-1$ in $\mathcal{F}$ is finite and fixed, and therefore a list of formulas generating all of them can be precalculated). By the $r-1$ decomposability of R, the following formula Ψ is equivalent to R.

$$\Psi = \bigwedge_{I=\{i_1,\ldots,i_r\},1\leq i_1\leq\cdots\leq i_r\leq n} \Phi_{\mathrm{pr}_I(R)}(x_{i_1},\ldots,x_{i_r})$$

∎

Corollary 1 *Let $\mathcal{F}$ be a class of formulas closed under a near-unanimity operation φ. Then, the following classes are polynomial-time learnable with proper equivalence queries: $\exists\text{-}\mathrm{Form}([\mathcal{F}])$, $\exists\forall\text{-}\mathrm{Form}([\mathcal{F}])$, $\exists\text{-}\mathrm{Form}_C([\mathcal{F}])$, and $\exists\forall\text{-}\mathrm{Form}_C([\mathcal{F}])$.*

7 Non-learnability Results

In the previous section we have used algebraic properties of relations to prove learnability results. More precisely, we used closure operations to show that Algorithm **GS** can efficiently learn some sets of formulas. We show in this section that the link between the complexity of learning classes of quantified formulas and closure operations is even tighter: the closure operations of a quantified formula determine the learnability complexity.

In order to establish this result we first introduce some standard algebraic terminology.

Definition 12 Let Γ be a set of relations over D. Define $\mathrm{Pol}(\Gamma)$ (the *polymorphisms* of Γ) to be the set of all operations, f, on D such that every relation in Γ is closed under f.

The polymorphisms of a set have a known structure, **(a)** they contain all the projection operations, i.e., all functions that simply return one of their arguments, and **(b)** they are closed under composition. Any set of operations satisfying these conditions is called a *clone* [29]. The set of all clones over some finite set D forms a lattice with top $\mathcal{O}_D$, where $\mathcal{O}_D$ is the clone containing all the operations over D, and with bottom $\mathcal{I}_D$ where $\mathcal{I}_D$ is the clone containing exactly the projection operations on D.

Definition 13 Let F be a set of operations on a set D. Define $\mathrm{Inv}(F)$ (invariants of F) to be the set of all relations over D which are closed under every element of F.

The mappings Pol and Inv establish a Galois connection between sets of relations and sets of operations [7, 20]. By making use of this Galois connection we can obtain considerable insight into the relationship between different learning problems.

The structure of the invariants of a set of operations has also been well studied (see [24] for example). Translating the known results into our notation, we obtain the following.

Theorem 8 *For every set of relations, Γ, over a finite set D, we have:*

1. $\exists\text{-}\,\mathrm{Rel}(\Gamma \cup \{\mathrm{eq}_D\}) = \mathrm{Inv}(\mathrm{Pol}(\Gamma))$
2. $\exists\forall\text{-}\,\mathrm{Rel}_C(\Gamma \cup \{\mathrm{eq}_D\}) = \exists\text{-}\,\mathrm{Rel}_C(\Gamma \cup \{\mathrm{eq}_D\}) = \mathrm{Inv}(\mathrm{Idem}(\mathrm{Pol}(\Gamma))))$.

Proof. Item **(1)** is stated in [24], using a rather different notation. It is straightforward to check that the smallest *co-clone* (as defined in [24]) containing Γ corresponds to $\exists\text{-}\,\mathrm{Rel}(\Gamma \cup \{\mathrm{eq}_D\})$.

Item **(2)** is rather immediate from part **(1)**. The first identity comes from the fact that every formula of the form $\forall x \Phi$ is equivalent to

$$\bigwedge_{d \in D} \Phi \begin{bmatrix} x \\ d \end{bmatrix},$$

where $\Phi\left[\begin{smallmatrix} x \\ d \end{smallmatrix}\right]$ is the formula obtained from Φ by replacing each occurrence of x with d.

The second identity is proved throughout a chain of identities. For every $i \in D$, let $R_i = \{\langle i \rangle\}$ be the unary relation that simulates constant i.

First, for every set Γ of relations over D we have

$$\exists\text{-}\operatorname{Rel}_C(\Gamma \cup \{\mathrm{eq}_D\}) = \exists\text{-}\operatorname{Rel}(\Gamma \cup \{R_0, \ldots, R_{|D|-1}\} \cup \{\mathrm{eq}_D\}).$$

To prove it, notice that for every $0 \le i \le |D| - 1$, R_i is obtained by $\mathrm{eq}_D(x, i)$. Thus

$$\exists\text{-}\operatorname{Rel}_C(\Gamma \cup \{\mathrm{eq}_D\}) \supseteq \exists\text{-}\operatorname{Rel}(\Gamma \cup \{R_0, \ldots, R_{|D|-1}\}\} \cup \{\mathrm{eq}_D\}).$$

For the converse consider any formula Φ in $\exists\text{-}\operatorname{Form}_C(\Gamma \cup \{\mathrm{eq}_D\})$. Consider the formula Ψ in $\exists\text{-}\operatorname{Form}(\Gamma \cup \{R_0, \ldots, R_{|D|-1}\}\} \cup \{\mathrm{eq}_D\})$ given by

$$\Psi = \exists x_0, \ldots, x_{|D|-1} \Phi' \wedge R_0(x_0) \wedge R_1(x_1) \wedge \cdots \wedge R_{|D|-1}(x_{|D|-1}),$$

where x_i $(i \in D)$ are auxiliary variables not in Φ and Φ' is the formula obtained taking Φ and replacing every occurrence of a constant i by its associated variable x_i. Formula Ψ is equivalent to Φ. So

$$\exists\text{-}\operatorname{Rel}_C(\Gamma \cup \{\mathrm{eq}_D\}) \subseteq \exists\text{-}\operatorname{Rel}(\Gamma \cup \{R_0, \ldots, R_{|D|-1}\} \cup \{\mathrm{eq}_D\}).$$

Furthermore, by part **1** of this theorem we have,

$$\begin{gathered}\exists\text{-}\operatorname{Rel}(\Gamma \cup \{R_0, R_1, \ldots, R_{|D|-1}\} \cup \{\mathrm{eq}_D\}) = \\ \operatorname{Inv}(\operatorname{Pol}(\Gamma \cup \{R_0, R_1, \ldots, R_{|D|-1}\} \cup \{\mathrm{eq}_D\}))\end{gathered}$$

Finally, notice that eq_D is preserved by all operations and $\{R_0, R_1, \ldots, R_{|D|-1}\}$ is preserved by exactly all the idempotent operations over D. Thus

$$\operatorname{Idem}(\operatorname{Pol}(\Gamma)) = \operatorname{Pol}(\Gamma \cup \{R_0, R_1, \ldots, R_{|D|-1}\} \cup \{\mathrm{eq}_D\}).$$

So, $\exists\text{-}\operatorname{Rel}_C(\Gamma \cup \{\mathrm{eq}_D\}) = \operatorname{Inv}(\operatorname{Idem}(\operatorname{Pol}(\Gamma)))$ follows from putting together the previous identities. ■

This result shows that the clone of polymorphisms of a basis Γ characterizes the expressive power of the class of formulas $\exists\text{-}\operatorname{Form}(\Gamma \cup \{\mathrm{eq}_D\})$. The next result (Lemma 9) shows that in many cases this clone also determines the learning complexity of this class in the PAC-prediction with membership queries learning model [2].

Lemma 9 *Let Γ and Γ_0 be sets of relations over a finite set D where Γ_0 is finite. Then:*

1. *If $\operatorname{Pol}(\Gamma) \subseteq \operatorname{Pol}(\Gamma_0)$, and $\exists\text{-}\operatorname{Form}(\Gamma \cup \{\mathrm{eq}_D\})$ is polynomially predictable with membership queries, so is $\exists\text{-}\operatorname{Form}(\Gamma_0)$.*
2. *If $\operatorname{Idem}(\operatorname{Pol}(\Gamma)) \subseteq \operatorname{Idem}(\operatorname{Pol}(\Gamma_0))$, and $\exists\text{-}\operatorname{Form}_C(\Gamma \cup \{\mathrm{eq}_D\})$ is polynomially predictable with membership queries, so is $\exists\text{-}\operatorname{Form}_C(\Gamma_0)$.*
3. *If $\operatorname{Idem}(\operatorname{Pol}(\Gamma)) \subseteq \operatorname{Idem}(\operatorname{Pol}(\Gamma_0))$, and $\exists\forall\text{-}\operatorname{Form}_C(\Gamma \cup \{\mathrm{eq}_D\})$ is polynomially predictable with membership queries, so is $\exists\forall\text{-}\operatorname{Form}_C(\Gamma_0)$.*

[2] We say "in many cases" because we have to add an additional hypothesis: the finiteness of one of the basis sets. This is the only place in the sequence of results leading up to the proof of the main result of this section in which this assumption is used. Furthermore, up to the present moment, there is no known counterexample showing that Lemma 9 is false without this new additional assumption. Thus, the study of the necessity of the finiteness assumption is an interesting open question.

Proof. We prove here item **1**. Proof of items **2** and **3** are analogous.

Every relation R in Γ_0 belongs to $\exists\text{-}\operatorname{Form}(\Gamma \cup \{\mathrm{eq}_D\})$ by Theorem 8. Then, given any formula Φ in $\exists\text{-}\operatorname{Form}(\Gamma_0)$, it is possible to replace every occurrence of a relation R in Γ_0 by a formula Ψ_R in $\exists\text{-}\operatorname{Form}(\Gamma \cup \{\mathrm{eq}_D\})$ with the same satisfying assignments. Since, Γ_0 is finite, the size of every replacement is bounded by a constant and, in consequence, the new formula has size polynomial in the size of the previous formula. Thus, let g be the function assigning to every formula Φ in $\exists\text{-}\operatorname{Form}(\Gamma_0)$ an associated formula in $\exists\text{-}\operatorname{Form}(\Gamma \cup \{\mathrm{eq}_D\})$ constructed as described above and let f and h be identity functions. Then, f, g, and h define a pwm-reduction from $\exists\text{-}\operatorname{Form}(\Gamma_0)$ to $\exists\text{-}\operatorname{Form}(\Gamma \cup \{\mathrm{eq}_D)\}$ ∎

The next result shows that the trivial relation eq_D can always be included without changing the complexity of learnability.

Lemma 10 *Let Γ be a set of relations over a finite domain D.*

1. *If $\exists\text{-}\operatorname{Form}(\Gamma)$ is polynomially predictable with membership queries, then so is $\exists\text{-}\operatorname{Form}(\Gamma \cup \{\mathrm{eq}_D\})$.*
2. *If $\exists\text{-}\operatorname{Form}_C(\Gamma)$ is polynomially predictable with membership queries, then so is $\exists\text{-}\operatorname{Form}_C(\Gamma \cup \{\mathrm{eq}_D\})$.*
3. *If $\exists\forall\text{-}\operatorname{Form}_C(\Gamma)$ is polynomially predictable with membership queries, then so is $\exists\forall\text{-}\operatorname{Form}_C(\Gamma \cup \{\mathrm{eq}_D\})$.*

Proof. We prove here item **1**. Proofs of items **2** and **3** are analogous. We give an algorithm to learn $\exists\text{-}\operatorname{Form}(\Gamma \cup \{\mathrm{eq}_D\})$ which uses the algorithm for learning $\exists\text{-}\operatorname{Form}(\Gamma)$ as a procedure:

We will only care about identity relations between free variables. Otherwise, they can be treated as multiple occurrences of the same variable. The algorithm starts by assuming that there exists an identity relation between every pair of free variables, this set of candidates is updated with every collision with the examples. This set can be regarded as an equivalence relation $(=)$ in the set of free variables, and the algorithm for identity-free formulas is used to learn $(\Phi/=)$, meaning the formula obtained taking only a representative for each set of variables constrained by identities. ∎

We are now in a position to establish the main result of this section, that the set of polymorphisms of a class of formulas determines its learning complexity.

Theorem 9 *Let Γ and Γ_0 be sets of relations over a finite set D where Γ_0 is finite. Then:*

1. *If $\operatorname{Pol}(\Gamma) \subseteq \operatorname{Pol}(\Gamma_0)$, and $\exists\text{-}\operatorname{Form}(\Gamma)$ is polynomially predictable with membership queries so is $\exists\text{-}\operatorname{Form}(\Gamma_0)$.*
2. *If $\operatorname{Idem}(\operatorname{Pol}(\Gamma)) \subseteq \operatorname{Idem}(\operatorname{Pol}(\Gamma_0))$, and $\exists\text{-}\operatorname{Form}_C(\Gamma)$ is polynomially predictable with membership queries, so is $\exists\text{-}\operatorname{Form}_C(\Gamma_0)$.*
3. *If $\operatorname{Idem}(\operatorname{Pol}(\Gamma)) \subseteq \operatorname{Idem}(\operatorname{Pol}(\Gamma_0))$, and $\exists\forall\text{-}\operatorname{Form}_C(\Gamma)$ is polynomially predictable with membership queries, so is $\exists\forall\text{-}\operatorname{Form}_C(\Gamma_0)$.*

Proof. Putting the previous results together, Theorem 9 is rather immediate. We only show item **1**. Items **2** and **3** are analogous.

First, by Lemma 10 if $\exists\text{-}\operatorname{Form}(\Gamma)$ is polynomially predictable with membership queries then so is $\exists\text{-}\operatorname{Form}(\Gamma \cup \mathrm{eq}_D)$. Since, $\operatorname{Pol}(\Gamma) \subseteq \operatorname{Pol}(\Gamma_0)$, by Lemma 9, $\exists\text{-}\operatorname{Form}(\Gamma_0)$ is also polynomially predictable with membership queries. This completes the proof of Theorem 9. ■

Theorem 9 allows us to classify the complexity of learning any set of quantified formulas over a finite basis set, based on the corresponding clone of polymorphisms of that basis.

Theorem 9 also holds for some other learning models, for example: exact learning with equivalence queries, PAC-learning, PAC-prediction, and the models obtained by adding membership queries to them. Since in this section we are interested in non-learnability results, we only state the theorem for the strongest model: PAC-prediction with membership queries.

The original aim of this paper was to extend the dichotomic classification for the learnability of Boolean formulas [9] to larger domains. Unfortunately, the 2-element domain is rather special. Whereas the clone lattice for the Boolean domain is countable and was fully identified by Post [26], much less is known about the clone lattice for larger domains. In fact, the clone lattice for $|D| \geq 3$, contains uncountably many clones [32]. This fact seems to indicate that it will be hard to find a complete classification for larger domains.

However, in the following sections we will show that by restricting the lattice of clones in certain ways, i.e., considering only minimal clones, or plain clones, we obtain a dichotomy.

7.1 Minimal clones

Atoms in the clone lattice are called *minimal clones*, that is, a clone $\mathcal{C}$ over the domain D is minimal if $\mathcal{C} \neq \mathcal{I}_D$ and $\mathcal{C}$, $\mathcal{I}_D$ are the only subclones of $\mathcal{C}$.

The learnability of minimal clones is completely classified .

Theorem 10 *(Dichotomy Theorem for minimal clones) Let Γ be a set of relations, such that $\operatorname{Pol}(\Gamma)$ is minimal. Then, if $\operatorname{Pol}(\Gamma)$ contains a majority operation or an affine operation then $\exists\forall\text{-}\operatorname{Form}_C(\Gamma)$ is polynomially learnable with improper equivalence queries, otherwise $\exists\text{-}\operatorname{Form}_C(\Gamma)$ is not polynomially predictable even with membership queries under cryptographic assumptions.*

In order to establish this result we shall make use of the following earlier result.

Theorem 11 *[9] Let Γ be any set of logical relations containing $[\overline{x} \vee \overline{y} \vee z]$ or $[x \vee y \vee \overline{z}]$, then $\exists\text{-}\operatorname{Form}_C(\Gamma)$ is not polynomially predictable with membership queries under cryptographic assumptions.*

Proof of Theorem 10

Every minimal clone is generated by one of the following operations [27, 29]:

1. an idempotent binary operation (which is not a projection); or
2. a majority operation; or
3. an affine operation; or
4. a semiprojection; or
5. a nonidentical unary operation.

where we say that a k-ary operation f ($k \geq 3$) is a semiprojection if there exists an index $i \in \{1, \ldots, k\}$ such that for all $d_1, d_2, \ldots, d_k \in D$ with $|\{d_1, d_2, \ldots, d_k\}| < k$ we have $f(d_1, \ldots, d_k) = d_i$, but f is not a projection.

The operations in cases **(2)** and **(3)** are particular cases of near-unanimity and coset generating operations respectively, so they are polynomially learnable, as shown above. Case **(4)** is also easy to study, since clearly all relations over two values are closed under any semiprojection, including the relation $[x \vee y \vee \bar{z}]$. Non-learnability of this class therefore follows from Theorem 11. Case **(5)** is simple because nonidentical unary operations are not idempotent. Since by Theorem 9, in order to study the learning complexity of existentially quantified formulas with constants, non-idempotent operations are non-relevant, the case is reduced to the case in which the clone contains only projections, namely, $\mathcal{I}_D$, which is intractable, since every relation is preserved by all projections, including the relation $[x \vee y \vee \bar{z}]$.

Case **(1)** is a bit more involved. We shall distinguish between affine and non-affine clones, making use of the following theorem due to Szendrei.

First, we have to introduce some terminology (see [29] for example). Every algebra $\mathbf{D} = (D; F)$ has a clone naturally associated with it: the clone $T(\mathbf{D})$ of term operations of $\mathbf{D}$, which is the clone generated by F, and is called the clone of $\mathbf{D}$. Two algebras with a common universe are called *term equivalent* iff they have the same term operations. Let $\mathbf{A_1} = (A, F_1)$ be an algebra. An algebra $\mathbf{A_2} = (A, F_2)$ is said to be a *reduct* of $\mathbf{A_1}$ iff $T(\mathbf{A_2}) \subseteq T(\mathbf{A_1})$. The full idempotent reduct of $\mathbf{A_1}$ is the algebra on A whose operations are the idempotent term operations of $\mathbf{A_1}$.

Theorem 12 *[30] A finite idempotent groupoid $(D; f)$ with minimal clone is term equivalent to an algebra $(D; x-y+z)$ for some elementary Abelian q-group (q is an odd prime) if and only if it has a minimal nonsingleton subgroupoid of cardinality greater than 2.*

As a corollary of this theorem we have that every clone generated by an idempotent non-affine binary operation contains (at least) one minimal subuniverse B of cardinality two. Let $B = \{0, 1\}$. A simple case analysis of the idempotent groupoids over B reveals that there exists only two idempotent groupoids over B which are not projections, namely $(B; \wedge)$ and $(B; \vee)$. Clearly, operation $\wedge$ preserves $[\overline{x} \vee \overline{y} \vee z]$ and, similarly, operation $\vee$ preserves $[x \vee y \vee \overline{z}]$. Thus, by Theorem 11 they are not learnable. This completes the proof. ∎

7.2 Plain Algebras

An algebra is called *plain* iff it is simple (see [20] for example) and has no nonsingleton proper subalgebras.

The learnability of plain algebras is completely classified .

Theorem 13 *(Dichotomy Theorem for plain algebras) Let* Γ *be a finite set of relations over* D*, such that* $(D; \mathrm{Pol}(\Gamma))$ *is plain. Then, if* $Pol(\Gamma)$ *contains a near-unanimity operation or an affine operation then* $\exists\forall\text{-}\mathrm{Form}_C(\Gamma)$ *is polynomially learnable with improper equivalence queries, otherwise* $\exists\text{-}\mathrm{Form}_C(\Gamma)$ *is not polynomially predictable even with membership queries under cryptographic assumptions.*

Note that this result contains and generalizes the dichotomy classification for the Boolean case [9].

In order to establish this result we shall make use of the fact that plain idempotent algebras of universes with cardinality larger than two are very restricted. They fall into three types identified by Szendrei (see Theorem 2.1 of [30]). First we need some notation.

For arbitrary mapping $g : D^n \to D$, we define a subset $g^{\square}$ of D^{n+1} by

$$g^{\square} = \{\langle x_1, \ldots, x_n, g(x_1, \ldots, x_n)\rangle : x_1, \ldots, x_n \in A\}.$$

For arbitrary operations f, g on D, f is said to *commute* with g iff f preserves $g^{\square}$. From the definition if follows that the commutativity of operations is a symmetric relation.

For a permutation group G acting on D we will denote by $\mathcal{O}_D(G)$ the clone of all idempotent operations on D commuting with every element of G.

In the description of plain idempotent algebras an important role is played by the *higher dimensional crosses*

$$X_n^d = \cup_{i=1}^n (\overbrace{D \times \cdots \times D}^{i-1} \times \{d\} \times D \times \cdots \times D), \ (n \geq 2),$$

where d is a fixed element of the set D. That is, X_n^d is the n-ary relation over D containing exactly all the tuples with at least one d. For $k \geq 2$ and $d \in D$, let $\mathcal{F}_k^d$ denote the clone of all idempotent operations preserving X_k^d.

Theorem 14 *[30] Every plain idempotent algebra* $(D; F)$ *with* $|D| \geq 3$ *is locally term equivalent to one of the following algebras:*

1. $(D; \mathcal{O}_D(G))$ *for a permutation group* G *acting on* D *such that every nonidentity member of* G *has at most one fixed point;*
2. *the full idempotent reduct of the module* $End_{K^{\mathbf{D}}}\mathbf{D}$ *for some vector space* $K^{\mathbf{D}} = (D; +, K)$ *over a division ring* K*.*
3. $(D; \mathcal{O}_D(G) \cap \mathcal{F}_k^0)$ *for some* k $(2 \leq k \leq |D|)$*, some element* $0 \in D$*, and a permutation group* G *acting on* D *such that* 0 *is the unique fixed point of every nonidentity member.*

In fact, as a consequence of this result we have:

Corollary 2 *For every plain idempotent algebra* $\mathbf{D} = (D; F)$*, with* $|D| \geq 3$*. The clone of* $\mathbf{D}$*,* $T(\mathbf{D})$ *contains either a near-unanimity operation or an affine operation.*

Proof. Consider any one of the cases of Theorem 14. In case **1**, $\mathbf{D}$ is locally equivalent to $(D; \mathcal{O}_D(G))$ for a permutation group G acting on D. For every element g of G, clearly $g^{\square} = \{(d, g(d)) : d \in D\}$ is a permutation. (or a 1-constraint) in the sense of [8]. It is straightforward to verify (see [8], for example) that this kind of constraints are preserved by a particular kind of majority operation $\mu : D^3 \to D$ called the "dual discriminator" and defined by

$$\mu(x, y, z) = \begin{cases} y & \text{if } y = z \\ x & \text{otherwise} \end{cases}$$

Thus, μ preserves every element in G and, in consequence, is in $T(\mathbf{D})$.

In case **2**, $\mathbf{D}$ is locally equivalent to the full idempotent reduct of the module ${}_{\mathrm{End}_{K^{\mathbf{D}}}}\mathbf{D}$ for some vector space $K^{\mathbf{D}} = (D; +, K)$ over a division ring K. By the definition of module, ${}_{\mathrm{End}_{K^{\mathbf{D}}}}\mathbf{D} = (D; +, -, 0, \mathrm{End}_{K^{\mathbf{D}}})$, where $(D, +, -, 0)$ is an abelian group. Therefore, $\varphi(x, y, z) = x - y + z$ is a idempotent term of ${}_{\mathrm{End}_{K^{\mathbf{D}}}}\mathbf{D}$ and, in consequence, φ belongs to $\mathbf{D}$.

In case **3**, $\mathbf{D}$ is locally equivalent to $(D; \mathcal{O}_D(G) \cap \mathcal{F}_k^0)$ for some k $(2 \leq k \leq |D|)$, some element $0 \in D$, and a permutation group G acting on D such that 0 is the unique fixed point of every nonidentity member. Consider now the near-unanimity operation $\varphi : D^{k+1} \to D$ of arity $k+1$ given by

$$\varphi(x_1, \ldots, x_{k+1}) = \begin{cases} x & \text{if } x_1 = \cdots = x_k = x \\ x & \text{if } x_1 = \cdots = x_{k-1} = x_{k+1} = x \\ \vdots & \vdots \\ x & \text{if } x_2 = \cdots = x_{k+1} = x \\ 0 & \text{otherwise} \end{cases}$$

Consider any element g in G. We will see that φ preserves $g^{\square}$. Let $t_1 = (d_1, g(d_1)), \ldots, t_{k+1} = (d_{k+1}, g(d_{k+1}))$ be $k+1$ tuples in $g^{\square}$. We have to consider two possibilities:

1. In the set $\{d_1, \ldots, d_{k+1}\}$ we have k identical elements with value d. Then the set $\{g(d_1), \ldots, g(d_{k+1})\}$ contains k elements identical to $g(d)$, then $\varphi(t_1, \ldots, t_{k+1}) = (d, g(d)) \in g^{\square}$.
2. Otherwise, since g is a permutation, the set $\{g(d_1), \ldots, g(d_{k+1})\}$ cannot have k identical elements. Then $g(t_1, \ldots, t_{k+1}) = (0, 0) \in g^{\square}$, since 0 is a fixed point in g.

Thus, φ belongs to $\mathcal{O}_D(G)$.

Now, we will see that φ preserves X_k^0. Let $t_1 \ldots, t_{k+1}$ be tuples in X_k^0. By construction of X_k^0 every tuple contains at least one 0, but since we have $k+1$ tuples, by the pigeon-hole principle, there exists a column with at least two 0's. Clearly, the result of applying φ to the elements in that column will be 0. In consequence $\varphi(t_1, \ldots, t_{k+1})$ contains at least one 0 and therefore belongs to X_k^0. Thus, φ belongs to $\mathcal{F}_k^0$. Finally then, φ belongs to $T(\mathbf{D})$. ■

On the other hand, every 2-element subalgebra is plain and it is known that there are more 2-element idempotent algebras than those that are listed in Szendrei's classification.

However, the Boolean case has been studied [9] using a different technique leading to a complete classification. As it is pointed out in Section 8, the learnable classes coincide precisely with relations closed under affine or near-unanimity operations. This completes the proof of Theorem 13.

8 Boolean Case revisited

In this section we obtain a much simpler proof for the already known results about the learnability of quantified Boolean formulas [9] by using the new perspective provided by the connection with clone theory described in this paper.

We start by introducing the dichotomy theorem for the learnability of quantified Boolean formulas with constants. The following definitions are from [9] but the notation has been slightly adapted for convenience:

A Boolean relation is *bijunctive* if it can be expressed as a CNF where every clause has at most 2 literals. A Boolean relation is said to be *k-weakly monotone* (resp. *k-weakly antimonotone*) ($k \geq 3$) if can be expressed as a CNF where every clause is either (i) the disjunction of at most k unnegated variables (resp. negated variables) or (ii) the disjunction of at most two literals with at most one negated (resp. unnegated) variable. A Boolean relation is *weakly monotone* (WM) (resp. *weakly antimonotone* (WA)) if it is k-ary weakly monotone (resp. k-ary weakly antimonotone) for some $k \geq 3$. Finally, a Boolean relation R is *linear* if it is logically equivalent to some system of equations over GF(2).

These definitions are extended in a natural way to sets of relations. We say, for example, that a set of relations Γ is bijunctive iff every relation in Γ is bijunctive. In [9] it is shown that under certain cryptographic assumptions, quantified formulas are polynomially learnable iff the basis is bijunctive, weakly monotone, weakly antimonotone or linear.

Theorem 15 *(Dichotomy Theorem for Learnability of Generalized Quantified Boolean Formulas) Let Γ be a finite set of Boolean relations. If Γ satisfies one of the conditions (a)-(d) below, then $\mathcal{C}_{\exists\forall\text{-}\mathrm{Form}_C(S)}$ is polynomially exactly learnable with improper equivalence queries. Otherwise, $\mathcal{C}_{\exists\text{-}\mathrm{Form}(S)}$ 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.

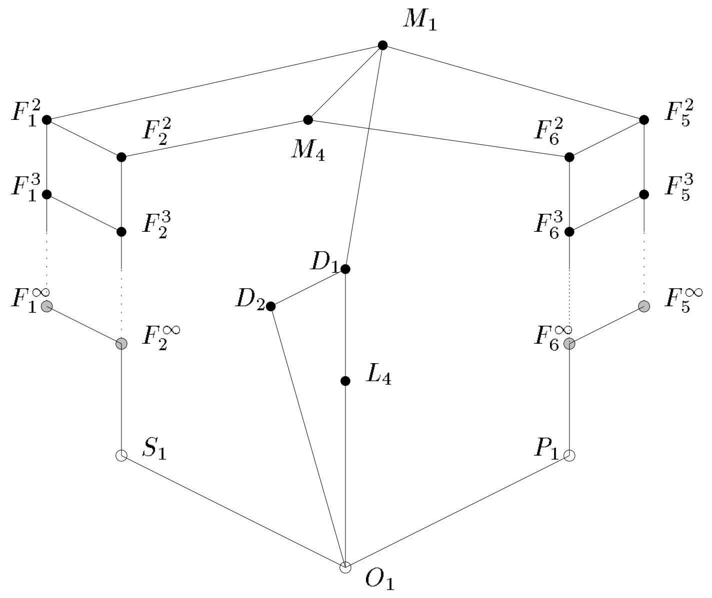

Figure 2: Post Lattice restricted to idempotent operations

Diagram 2 represents the idempotent Post lattice, that is, the lattice of clones on a 2-element set [26] restricted to idempotent operations. Nodes in Figure 2 are labeled according to Post's notation (see [24] for a simple description of the lattice). Grey nodes denote clones generated by an infinite set of relations. Black nodes denote clones containing a near-unanimity or an affine function. The following table establishes the correspondence between these clones and the learnable classes in the Boolean case.

bijunctive	$\varphi_{D_2}(x,y,z) = xy \vee yz \vee zx$
k-ary WM	$\varphi_{F_2^k}(x_1,\ldots,x_{k+1}) = \bigwedge_{j=1}^{k+1}(x_1 \vee .. \vee x_{j-1} \vee x_{j+1} \vee .. \vee x_{k+1})$
k-ary WA	$\varphi_{F_6^k}(x_1,\ldots,x_{k+1}) = \bigvee_{j=1}^{k+1}(x_1 \wedge .. \wedge x_{j-1} \wedge x_{j+1} \wedge .. \wedge x_{k+1})$
linear	$\varphi_{L_4}(x,y,z) = x \oplus y \oplus z$

Every entry in the table, consisting of a class of relations C and an operation φ_X, should be interpreted in the following way: a basis S belongs to the class C iff $\varphi_X \in \mathrm{Pol}(S)$. Furthermore, operation φ_X generates clone X.

Clearly, φ_{L_4} is an affine operation, and φ_{D_2}, $\varphi_{F_2^k}$, and $\varphi_{F_6^k}$ are near-unanimity operations. So, the learnability of the associated classes follows directly from Theorems 5 and 7. As the lattice is ordered from bottom to top according to inclusion, the remainder of the black nodes also contain a near-unanimity or an affine operation inherited from some ancestor.

The remainder of the clones are depicted as white nodes in figure 2.

Let us now analyze them. Consider clone P_1, which is generated by the conjunction operation. It is known that every Horn formula is closed under conjunction and in particular so is $[\overline{x} \vee \overline{y} \vee z]$. On the other hand, clone S_1 is generated by the disjunction operation, so it preserves all anti-Horn formulas, in particular $[x \vee y \vee \overline{z}]$. Therefore, every clone generated by a finite set of relations and not containing any near-unanimity or any affine operation preserves either $[\overline{x} \vee \overline{y} \vee z]$ or $[x \vee y \vee \overline{z}]$ so, by Theorem 11, the associated set of quantified formulas is not polynomially predictable with membership queries under cryptographic assumptions. This simple reasoning gives us an alternative proof in place of the involved case analysis originally given in [9].

References

[1] D. Angluin. Queries and Concept Learning. *Machine Learning*, 2:319–342, 1988.

[2] D. Angluin, M. Frazier, and L. Pitt. Learning Conjunctions of Horn Clauses. *Machine Learning*, 9:147–164, 1992.

[3] D. Angluin and M. Kharitonov. When won't Membership Queries help. *Journal of Computer and System Sciences*, 50:336–355, 1995.

[4] K.A. Baker and A.F. Pixley. Polynomial Interpolation and the Chinese Remainder Theorem for Algebraic Systems. *Math. Z.*, 1975.

[5] N.H. Bshouty. Exact Learning Boolean Functions via the Monotone Theory. *Information and Computation*, pages 146–153, November 1995.

[6] Zhixiang Chen and Steven Homer. Learning counting functions with queries. *Theoretical Computer Science*, 180(1–2):155–168, 10 June 1997.

[7] P.M. Cohn. *Universal Algebra*. Harper and Row, 1965.

[8] M.C. Cooper, D.A. Cohen, and P.G. Jeavons. Characterizing Tractable Constraints. *Artificial Intelligence*, 65:347–361, 1994.

[9] V. Dalmau. A Dichotomy Theorem for Learning Quantified Boolean Formulas. *Machine Learning*, 35(3):207–224, 1999.

[10] T. Feder and M.Y. Vardi. The Computational Structure of Monotone Monadic SNP and Contraint Satisfaction: A Study through Datalog and Group Theory. *SIAM J. Computing*, 28(1):57–104, 1998.

[11] D. Helmbold, R. Sloan, and M. Warmuth. Learning Nested Differences of Intersection-Closed Concept Classes. *Machine Learning*, 5(2):165–196, 1990.

[12] M.C. Hoffmann. *Group-Theoretic Algorithms and Graph Isomorphism*, volume 136 of *Lecture Notes in Computer Science*. Springer-Verlag, 1982.

[13] Jeffrey C. Jackson. An Efficient Membership-query Algorithm for Learning DNF with respect to the Uniform Distribution. *Journal of Computer and System Sciences*, 55(3):414–440, December 1997.

[14] P. Jeavons, D. Cohen, and M.C. Cooper. Constraints, Consistency and Closure. *Artificial Intelligence*, 101:251–265, 1998.

[15] P. Jeavons, D. Cohen, and M. Gyssens. A Unifying Framework for Tractable Constraints. In *1st International Conference on Principles and Practice of Constraint Programming, CP'95, Cassis (France), September 1995*, volume 976 of *Lecture Notes in Computer Science*, pages 276–291. Springer-Verlag, 1995.

[16] P. Jeavons, D. Cohen, and M. Gyssens. Closure Properties of Constraints. *Journal of the ACM*, 44(4):527–548, July 1997.

[17] P. Jeavons and M. Cooper. Tractable Constraints on Ordered Domains. *Artificial Intelligence*, 79:327–339, 1996.

[18] Michael Kearns and Leslie Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. *Journal of the ACM*, 41(1):67–95, January 1994.

[19] W. Maass and G. Turán. On Learnability and Predicate Logic. In *Bar-Ilan Symposium on the Foundations of Artificial Intelligence, BISFAI'95*, pages 75–85, 1995.

[20] R.N. McKenzie, G.F. McNulty, and W.F. Taylor. *Algebras, Lattices and Varieties*, volume 1. Wadsworth and Brooks, 1987.

[21] S. Muggleton, editor. *Inductive Logic Programming*. Academic Press, 1992.

[22] S. Muggleton and L. De Raedt. Inductive Logic Programming: Theory and Methods. *Journal of Logic Programming*, 19/20:7:629–679, 1994.

[23] B.K. Natarajan. On Learning Boolean Functions. In *9th Annual ACM Symposium on Theory of Computing, STOC'87*, pages 296–304, 1987.

[24] N. Pippenger. *Theories of Computability*. Cambridge University Press, 1997.

[25] L. Pitt and M. Warmuth. Prediction-Preserving Reducibility. *Journal of Computer and System Sciences*, 41:430–467, 1990.

[26] E.L. Post. *The Two-Valued Iterative Systems of Mathematical Logic*, volume 5 of *Annals of Mathematics Studies*. Princeton, N.J, 1941.

[27] I.G. Rosenberg. Minimal Clones I:The Five Types. In *Proceedings Conference Szeged 1983, Colloq. Math. Soc. Janos Bolyai 43*, Lectures in Universal Algebra, pages 405–427. North Holland, 1986.

[28] T.J. Schaefer. The Complexity of Satisfiability Problems. In *10th Annual ACM Symposium on Theory of Computing*, pages 216–226, 1978.

[29] A. Szendrei. *Clones in Universal Algebra*, volume 99 of *Seminaires de Mathématiques Supéreiores*. University of Montreal, 1986.

[30] A. Szendrei. Idempotent algebras with restrictions in subalgebras. *Acta Sci. Math.*, 51:251–268, 1987.

[31] L. Valiant. A Theory of the Learnable. *Comm. ACM*, 27(11):1134–1142, 1984.

[32] Yu.I. Yanov and A.A. Muchnik. On the existence of k-valued closed classes without a finite basis. *Dokl. Akad. Nauk SSSR*, 127:44–46, 1959. In russian.