Learning Intersection-Closed Classes with Signatures

Andrei Bulatov [a] Hubie Chen [b] Víctor Dalmau [b]

[a] *Simon Fraser University, Burnaby, Canada*

[b] *Departament de Tecnologia, Universitat Pompeu Fabra, Barcelona, Spain*

Abstract

Intersection-closed classes of concepts arise naturally in many contexts and have been intensively studied in computational learning theory. In this paper, we study intersection-closed classes that contain the concepts invariant under an operation satisfying a certain algebraic condition. We give a learning algorithm in the exact model with equivalence queries for such classes. This algorithm utilizes a novel encoding scheme which we call *signature*.

1 Introduction

Intersection-closed classes of concepts arise naturally in many contexts and have been intensively studied in computational learning theory. Examples of intersection-closed classes include axis-parallel n-dimensional rectangles, k-CNF boolean functions, and systems of linear equations; see [4] for more examples and references. A known method for learning intersection-closed classes is the *closure algorithm* [4,18–20,27]. The idea behind the closure algorithm is to predict according to the intersection of all concepts containing the known positive examples.

In this paper, we study intersection-closed classes that contain the concepts invariant under a certain operation. Roughly speaking, a concept is invariant under an operation f if for any elements of the concept, applying f to the elements yields another element in the concept. We denote by $\mathrm{Inv}(f)$ the set of all concepts invariant under f.

Email addresses: abulatov@cs.sfu.ca (Andrei Bulatov), hubie.chen@upf.edu (Hubie Chen), victor.dalmau@upf.edu (Víctor Dalmau).

Perhaps the concept class that best illustrates this notion is that of affine systems over a given (finite) field F. This class contains all concepts that can be expressed as the set of all solutions of a system of linear equations $A \cdot \mathbf{x} = \mathbf{b}$. It is an easy exercise to show that any such concept is invariant under the operation $g(x, y, z) = x - y + z$ of the field. Indeed, for every three vectors $\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}$ satisfying $A \cdot \mathbf{x} = \mathbf{b}$, we have that

$$A \cdot g(\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}) = A \cdot (\mathbf{x_1} - \mathbf{x_2} + \mathbf{x_3}) = A \cdot \mathbf{x_1} - A \cdot \mathbf{x_2} + A \cdot \mathbf{x_3} = \mathbf{b}.$$

Another example that fits in this framework is the class of generalized quantified formulas. The complexity of testing satisfiability of generalized quantified formulas, also called quantified constraint satisfaction problems, has been studied in recent years; see for example [5,9,11,10]. Its learning complexity is studied in [15,16].

It has been proved [16] that the closure algorithm efficiently learns $\mathrm{Inv}(f)$ when f is a *near-unanimity operation*, an operation satisfying the identities

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

(Note that these identities state that if all but at most one of the inputs to f are equal, then the majority input is the output of f.) This result implies the learnability of quantified 2-CNF formulas [15]. It has also been proved [16] that the closure algorithm efficiently learns $\mathrm{Inv}(f)$ when f is a *coset-generating operation*, an operation equal to $x \cdot y^{-1} \cdot z$ where $\cdot$ and $^{-1}$ are the operations of a finite group. Note that any coset-generating operation f satisfies the identities

$$f(x, y, y) = f(y, y, x) = x.$$

This latter result subsumes the learnability of systems of linear equations over a finite field.

In this paper, we unify and generalize these two results. We consider classes of the form $\mathrm{Inv}(f)$ when $f : A^k \to A$ is a *generalized majority-minority operation* on a set A, that is, an operation such that, for all $a, b \in A$, either

$$f(x, y, .., y, y) = f(y, x, .., y, y) = \cdots = f(y, y, .., y, x) = y \text{ for all } x, y \in \{a, b\},$$

or

$$f(x, y, .., y, y) = f(y, y, .., y, x) = x \text{ for all } x, y \in \{a, b\}.$$

As is easily seen, previously studied near-unanimity and coset-generating operations correspond to generalized majority-minority operations, for which either $\{a, b\}$ satisfying the second condition never occurs (near-unanimity operations), or the operation is ternary and $\{a, b\}$ satisfying the first condition never occurs (coset-generating operations). We prove that the closure algorithm learns such classes in polynomial time, and therefore $\mathrm{Inv}(f)$, where f is

a generalized majority-minority operation, is learnable in the exact model with equivalence queries. However, the encoding produced by the closure algorithm, a generating set, is not known to be polynomial time evaluable. So we introduce a more sophisticated encoding for concepts invariant under a generalized majority-minority operation, which we call signature. For this encoding, we prove polynomial-time evaluability, that is, we can decide in polynomial time whether or not an example belongs to a concept. We then give an algorithm that exactly learns, in polynomial time, $\mathrm{Inv}(f)$ encoded with signatures.

2 Learning Preliminaires: The Closure Algorithm

In this paper, the model of learning that we use is exact learning with equivalence queries, defined by Angluin [1]. Note that in this model, the learning algorithm is relatively unpowerful, and so our positive learnability result implies learnability in "more powerful" models such as the PAC model [32], exact learning with equivalence and membership queries [1], the PAE model [6], and on-line learning [1,26]; these models are among the most studied in computational learning theory.

In our setting a concept c from a concept class C is merely a subset of a domain space X, paired with an encoding of c. Throughout this paper we shall assume that the domain space X contains all tuples over a given (finite) domain A, that is $X = A^*$. We shall denote the elements of X with boldface letters $\mathbf{x}, \mathbf{y}, \ldots$.

A learning algorithm Alg has the goal of identifying a target concept t. It may make any number of queries or requests to a teacher or oracle, who is commited to answer according to t, although not necessarily in a collaborative way. In the exact model with equivalence queries [1] the learner supplies the oracle with a hypothesis h and the oracle either says "equivalent", or returns a counterexample $\mathbf{x} \in h \triangle t$ (here, $\triangle$ denotes the symmetric difference). If the provided hypothesis h does not belong to the concept class C then we speak of *improper* equivalence queries. We say that Alg learns a concept class C if for every target concept t, Alg halts with an output concept v equivalent to c. Alg runs in polynomial time if its running time is bounded by a polynomial on the size of the encoding of t and the size of the largest counterexample. We shall say that a concept class C is *polynomially learnable* with equivalence queries if there exists a learning algorithm that learns C and runs in polynomial time. We also say that a concept class C is *polynomially evaluable* if there exists an algorithm that, given a concept $c \in C$ and an example $\mathbf{x} \in X$, decides whether or not $\mathbf{x} \in c$ and runs in polynomial time.

Definition 1 *A concept class C on domain X is* intersection closed *if for all*

two concepts $c_1, c_2 \subseteq X$ in C, its intersection $c_1 \cap c_2$ also belongs to C.

Examples of intersection-closed concept classes include axis-parallel n-dimensional rectangles, k-CNF boolean functions, subspaces of a linear space, and integer lattices.

There is a canonical algorithm for learning intersection-closed classes with equivalence queries, which has been called the *closure algorithm*: the hypothesis supplied by the algorithm is always the smallest concept c containing the set c' of all the counterexamples seen so far. This concept, which we denote as $\langle c' \rangle$, can be defined as $\bigcap_{d \in C, c' \subseteq d} d$. Due to the way in which the algorithm is defined, at any point of execution, the concept of the learner is a subset of the target concept. Thus, any conterexample provided by the oracle must be positive, that is, any counterexample belongs to the target concept.

3 Relations invariant under an operation

We shall focus on a particular type of intersection-closed classes.

We need to introduce some notation. Let A be a finite set. A k-ary *relation* (with $k \geq 0$ an integer) on A is a subset of A^k where k is called the *arity* of the relation. The set $\{1, \ldots, k\}$ is referred to as the set of *indices* or *coordinates* of R.

Definition 2 *Let A be a finite set, let $f : A^n \to A$ be an n-ary operation on A and let $R \subseteq A^m$ be an m-ary relation on A. We say that f* preserves *R (or f is a* polymorphism *of R, or R is* invariant *under f) if, for any $\mathbf{a_1} = (a_{11}, \ldots, a_{m1}), \ldots, \mathbf{a_n} = (a_{1n}, \ldots, a_{mn}) \in R$, the tuple $f(\mathbf{a_1}, \ldots, \mathbf{a_n})$ defined as $(f(a_{11}, \ldots, a_{1n}), \ldots, f(a_{m1}, \ldots, a_{mn}))$ belongs to R.*

We emphasize that, when speaking of a function f preserving a relation $R \subseteq A^m$, the function f is defined on the set A and not on A^m, although we extend the action of f to A^m (for all $m \geq 1$) by applying it coordinate-wise (as described in the definition).

Example 1 *Let R_1 be the binary relation over the boolean domain $A = \{0, 1\}$ that is given by*

$$R_1 = \{(0, 1), (1, 0), (1, 1)\}$$

That is, $R_1(x, y)$ is equivalent to $x \vee y$.

Let $m : \{0, 1\}^3 \to \{0, 1\}$ be the ternary operation on $\{0, 1\}$ that returns the

majority of its arguments. That is,

$$m(x,y,z) = \begin{cases} x \ \textit{if } x = y \\ z \ \textit{otherwise} \end{cases}$$

It is not difficult to verify that m is a polymorphism of R_1. We only need to check that for every three (not necessarily different) tuples in R_1, for example $(0,1),(1,0),(1,1)$, the tuple obtained by applying m component-wise, which in our example is $(m(0,1,1),m(1,0,1)) = (1,1)$, belongs to R_1.

Indeed, it is easy to see that m is also a polymorphism of R_2 and R_3 where $R_2(x,y) \equiv \overline{x} \vee y$ and $R_3(x,y) \equiv \overline{x} \vee \overline{y}$.

We are now in a position to define the family of concept classes that we are studying in this paper.

Definition 3 *Let A be a finite set, and let $f : A^k \to A$ be any function on A. We denote by* $\mathrm{Inv}(f)$ *the set containing all relations invariant under f.*

We shall slightly abuse the notation and we shall speak of $\mathrm{Inv}(f)$ as a concept class. Properly speaking, in order for $\mathrm{Inv}(f)$ to define a concept class it is necessary to associate an encoding to each relation in $\mathrm{Inv}(f)$. The particular encoding used might change from application to application.

In the closure algorithm relations are encoded by means of *generating sets*. For a set R' of n-tuples, $\langle R' \rangle_f$ denotes the smallest relation R such that $R' \subseteq R$ and R is invariant under f. This relation can also be represented as the intersection of all n-ary relations containing R' and invariant under f. We refer to this relation as the *relation generated by* R', and R' is said to be a generating set of R. Then, a relation R is encoded by a generating set. In the algorithm presented in this paper we shall use a different encoding called *signatures*.

Observe that the domain space of $\mathrm{Inv}(f)$ is precisely the set of all tuples with elements of A. Furthermore every concept in $\mathrm{Inv}(f)$ is bound to have examples (tuples) of the same length. It is easy to prove that $\mathrm{Inv}(f)$ is indeed intersection-closed. In the following we shall present some concept classes that fit into this framework.

3.1 Affine systems

Let F be an arbitrary finite field. Every system of linear equations over F with variables say, $x_1, \ldots, x_n$, encodes a concept which is constituted by all solutions of the system. We view every solution as an n-ary tuple, so the solution space is an n-ary relation R.

As mentioned in the introduction, the operation $g(x, y, z) = x - y + z$ is a polymorphism of R. In fact, the converse can also be shown: if R is invariant under g then it is the solution space of a certain system of linear equations.

The operation $x - y + z$ is called an *affine* operation. A similar operation $x \cdot y^{-1} \cdot z$, where $\cdot$ and $^{-1}$ are operations of a group is called a *coset-generating operation*. Note that every affine operation is a coset-generating operation arising from an Abelian group.

In [16] it is shown that $\mathrm{Inv}(g)$ is polynomially exactly learnable with equivalence queries by the closure algorithm.

3.2 Generalized Quantified Formulas

We use $[n]$ to denote the set containing the first n positive integers, that is, $\{1, \ldots, n\}$. Let $V = \{x_1, x_2, \ldots\}$ be a countably infinite set of variables and let A be a finite set.

Definition 4 *[16] Let A be any finite set and let $\Gamma = \{R_1, R_2, \ldots\}$ be any set of relations over A, where each R_i has arity k_i. (The notation R_i is used for both the relation and its symbol.)*

The set of quantified generalized formulas over the basis Γ*, which is denoted by* $\forall\exists$-Form(Γ)*, is the smallest set of first-order formulas such that:*

- *For all $R \in \Gamma$ of arity k, $R(y_1, \ldots, y_k) \in \forall\exists$-Form($\Gamma$) where $y_i \in V$ for $1 \leq i \leq k$.*
- *For all $\Phi, \Psi \in \forall\exists$-Form($\Gamma$), $\Phi \wedge \Psi \in \forall\exists$-Form($\Gamma$).*
- *For all $\Phi \in \forall\exists$-Form(Γ) and for all $x \in V$, $\exists x\Phi \in \forall\exists$-Form($\Gamma$).*
- *For all $\Phi \in \forall\exists$-Form(Γ) and for all $x \in V$, $\forall x\Phi \in \forall\exists$-Form($\Gamma$).*

Each formula Φ defines a relation R_Φ 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 $i_1 < i_2 < \cdots < i_m$; we define

$$R_\Phi = \{(a_1, a_2, \ldots, a_m) : x_{i_j} \to a_j \text{ satisfies } \Phi\}$$

Example 2 *Consider the problem of learning quantified 3-CNFs, which are formulas formed by a quantified conjunction of clauses with three literals per clause.*

Every such formula can be expressed as a formula in $\forall\exists$-Form(Γ) *with the set*

of logical relations $\Gamma = \{R_0, R_1, R_2, R_3\}$*, defined by:*

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

As an example, let Φ *be following formula:*

$$\exists x_1 \exists x_2 \forall x_3 \exists x_4 \forall x_5 R_1(x_3,x_4,x_5) \wedge R_1(x_2,x_2,x_2) \wedge R_1(x_6,x_5,x_4) \\ \wedge R_2(x_6,x_7,x_1) \wedge R_3(x_2,x_8,x_4) \wedge R_3(x_1,x_1,x_1)$$

Φ *is a formula in* $\forall\exists$-Form$\,\Gamma$ *over the free variables* x_6*,* x_7*, and* x_8*.* R_Φ *contains exactly all the assignments over these variables satisfying* Φ*:*

$$R_\Phi = \{(0,0,0),(0,1,0),(1,0,0)\}$$

In what follows we shall see that quantified generalized formulas fit in this framework. First, recall that an operation f on a set A is said to be *idempotent* if $f(a,\ldots,a) = a$ for all $a \in A$.

Lemma 1 *Let* A *be a finite set, let* f *be any idempotent operation on* A *and let* $\Gamma \subseteq \mathrm{Inv}(f)$ *be any set of relations invariant under* f*. Then every relation represented by a formula from* $\forall\exists$-Form(Γ) *is also invariant under* f*.*

Proof. We shall prove that for every formula Ψ in $\forall\exists$-Form(Γ), R_Ψ is invariant under f. It is proven by structural induction on Ψ (see also [16,5]). Cases $\Psi = \Phi_1 \wedge \Phi_2$ and $\Psi = \exists x \Phi$ are well-known, see e.g. [17]. Thus the only case to consider is when Ψ is obtained by means of universal quantification $\Psi = \forall x \Phi$. In this case we use the fact that Ψ is equivalent to

$$(\exists x \;\; \Phi \wedge a_1'(x)) \wedge \cdots \wedge (\exists x \;\; \Phi \wedge a_r'(x)),$$

where $\{a_1,\ldots,a_r\}$ are the elements of A and a_i' is the singleton relation $\{(a_i)\}$, for $1 \leq i \leq r$. Observe that, as we introduce singleton relations, they must be invariant under f. This is provided by the idempotency of f. $\Box$

3.3 Near-Unanimity and Coset-Generating Operations

Let A be a finite set. An operation $f : A^k \to A$ with $k \geq 3$ is called a *near-unanimity* operation if

$$f(x,y,\ldots,y) = \cdots = f(y,\ldots,y,x) = y$$

for all $x, y \in A$. In this paper, it will be convenient for us to call such an operation *generalized majority*.

Recall that a ternary operation m is called coset generating if $m(x, y, z) = x \cdot y^{-1} \cdot z$, where $\cdot$ and $^{-1}$ are the operations of a certain finite group.

The complexity of learning $\mathrm{Inv}(f)$ (encoded with generating sets) is addressed in [16].

Theorem 1 ([16]) *Let f be a near-unanimity or a coset generating operation. Then the closure algorithm learns exactly* $\mathrm{Inv}(f)$ *(encoded with generating sets) in polynomial time.*

It is worth remarking that if f is a near-unanimity or a coset-generating operation then every relation in $\mathrm{Inv}(f)$ has a generating set of size polynomial in the arity n of the relation.

Consequently, the learning algorithm in [16] learns $\mathrm{Inv}(f)$ with a running time that is polynomial in n. Hence, the same algorithm can be used to learn efficiently $\mathrm{Inv}(f)$ under any other encoding. In particular we have:

Corollary 1 *The class of systems of linear equations over a finite field F are polynomially exactly learnable with (improper) equivalence queries.*

Corollary 2 *Let Γ be a set of relations on a finite set A. If $\Gamma \subseteq \mathrm{Inv}(f)$ where f is either a near-unanimity or coset generating operation, then $\forall\exists$-Form(Γ) is polynomially exactly learnable with (improper) equivalence queries.*

Example 3 *A quantified 2-CNF formula is a $\forall\exists$-Form($\{R_1, R_2, R_3\}$) formula where R_1, R_2, R_3 are as defined in Example 1. Furthermore it is easy to observe that the operation m defined also in Example 1 is a near-unanimity operation. Consequently, we can infer that $\forall\exists$-Form($\{R_1, R_2, R_3\}$), and hence the family of quantified 2-CNFs, is polynomially exactly learnable with equivalence queries. Note that in this case the arity of the target concept is just the number of variables in a CNF.*

4 Results

In this section we introduce generalized majority-minority operations and signatures, and state the main results.

We start with the main definition.

Definition 5 *Let A be a finite set. An operation $f : A^k \to A$ with $k \geq 3$ is a* generalized majority-minority (GMM) operation *if for all $a, b \in A$, either*

$$f(x, y, \ldots, y) = f(y, x, \ldots, y) = \cdots = f(y, \ldots, y, x) = y \text{ for } x, y \in \{a, b\} \quad (1)$$

or

$$f(x, y, \ldots, y, y) = f(y, y, \ldots, y, x) = x \text{ for } x, y \in \{a, b\}. \quad (2)$$

Let us fix a GMM operation on a set A. A pair $a, b \in A$ is said to be a *majority* pair if f on a, b satisfies (1). It is said to be a *minority* pair if f satisfies (2).

Observe that a GMM operation that contains only majority pairs is a near-unanimity operation and that, indeed, every near-unanimity operation can be seen as such. Also, every coset-generating operation is a ternary GMM operation with only minority pairs.

Example 4 *It is easy to see that the ternary operation f defined as follows is a GMM operation. Let $A = \{a, b, c\}$, and let f be an operation on A such that*

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

if $x, y, z \in \{a, b\}$,

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

if $x, y, z \in \{a, c\}$ or $x, y, z \in \{b, c\}$, and $f(x, y, z) = x$ if $\{x, y, z\} = A$. Pair $\{a, b\}$ is minority with respect to f, and $\{a, c\}, \{b, c\}$ are majority pairs.

4.2 Signatures

We study the learning complexity of $\mathrm{Inv}(f)$ when f is a generalized majority-minority operation. Observe that in order to formulate the question precisely we have to associate to $\mathrm{Inv}(f)$ some representation scheme. A natural choice is to use generating sets. That is, we could represent a relation R in Γ by means of a subrelation R' of R that generates it. Indeed, we are able to prove, see Section 5.1, that the closure algorithm polynomially exactly learns $\mathrm{Inv}(f)$ encoded with generating sets. However, we hit a problem: it is still not known if a generating set can be used to predict new examples efficiently. Formally, it is not known whether it is possible to decide in polynomial time whether

a given tuple belongs to the relation generated by a given set of tuples (with respect to f). In order to overcome this difficulty we introduce a new encoding scheme for $\mathrm{Inv}(f)$, called *signatures*.

Before defining signatures we need to introduce a little bit of notation. Let A be a finite set and f be a GMM operation of arity k. Let also R be an n-ary relation on a finite set A. For a set $I = \{i_1, \dots, i_m\} \subseteq [n]$, $1 \leq i_1 < \dots < i_m \leq n$, we define the projection of a tuple $\mathbf{a} = (a_1, \dots, a_n)$ over I to be the tuple $\mathrm{pr}_I \mathbf{a} = (a_{i_1}, \dots, a_{i_m})$, and the projection of R over I to be the relation $\mathrm{pr}_I R = \{\mathrm{pr}_I \mathbf{a} \mid \mathbf{a} \in R\}$.

In the following definition we exploit the two properties of a k-ary GMM operation f on a set A. Let us consider first two extreme cases. Suppose that all pairs from A are majority. Then f satisfies the identities $f(x, y, \dots, y, y) = f(y, x, \dots, y, y) = \dots = f(y, y, \dots, y, x) = y$ for all $x, y \in A$, and therefore is a near-unanimity operation. It is well-known [22] that if n-ary relations R and S are invariant under a near-unanimity operation and, for any $I \subseteq [n]$ with $|I| = k - 1$, we have $\mathrm{pr}_I R = \mathrm{pr}_I S$, then $R = S$. Thus a relation invariant under f is uniquely determined by the collection of its $k-1$-ary projections.

Now suppose that all pairs from A are minority, and hence f satisfies the identities $f(x, y, \dots, y, y) = f(y, y, \dots, y, x) = x$ for all $x, y \in A$. In this case for any relation R invariant under f the following property of *rectangularity* holds: for any $i \in [n]$ and any $a, b, a_1, \dots a_{i-1}, b_1, \dots, b_{i-1} \in A$ if $\mathbf{a} = (a_1, \dots, a_{i-1}, a), \mathbf{b} = (a_1, \dots, a_{i-1}, b), \mathbf{c} = (b_1, \dots, b_{i-1}, a) \in \mathrm{pr}_{[i]} R$, then $\mathbf{d} = (b_1, \dots, b_{i-1}, b) \in \mathrm{pr}_{[i]} R$. Indeed, it is not hard to see that $\mathbf{d} = f(\mathbf{c}, \mathbf{a}, \dots, \mathbf{a}, \mathbf{b})$. Therefore, R is uniquely determined by the collection of such rectangularity pairs a, b for each i, which have a common extension onto coordinates $1, \dots, i-1$. (We actually also need to know at least one such extension for every rectangularity pair.)

As in the general case both majority and minority pairs may occur, we need a mixture of these two types of representations.

A *signature* of arity n is a triple $(\mathsf{Pro}, \mathsf{Rec}, \mathsf{Wit})$ where

(A1) Pro is a collection of pairs $(I, \mathbf{a})$ where I is a subset of $[n]$ of cardinality at most $k-1$ and $\mathbf{a}$ is a tuple on A whose arity matches the cardinality of I.
(A2) Rec is a collection of triples (i, a, b) where $1 \leq i \leq n$ and $a, b \in A$ is a minority pair.
(A3) Wit is a collection of n-ary tuples on A.

Furthermore Pro, Rec and Wit must satisfy the following conditions:

(B1) For every $(I, \mathbf{a}) \in \mathsf{Pro}$, Wit contains some tuple $\mathbf{x}$ such that $\mathrm{pr}_I \mathbf{x} = \mathbf{a}$.
(B2) For every $(i, a, b) \in \mathsf{Rec}$, Wit contains some tuples $\mathbf{a}, \mathbf{b} \in R$ such that

$\mathrm{pr}_{[i-1]}\,\mathbf{a} = \mathrm{pr}_{[i-1]}\,\mathbf{b}$, $\mathrm{pr}_i\,\mathbf{a} = a$, and $\mathrm{pr}_i\,\mathbf{b} = b$.

(B3) $|\mathsf{Wit}| \leq |\mathsf{Pro}| + 2|\mathsf{Rec}|$.

In the notation $(\mathsf{Pro}, \mathsf{Rec}, \mathsf{Wit})$, Pro stands for 'projections', Rec for 'rectangularities', and Wit for 'witnesses'. We shall denote by Sig_n the set of all signatures of arity n and by Sig the set $\bigcup_{n\geq 0} \mathrm{Sig}_n$.

Let R be an n-ary relation and let $(\mathsf{Pro}, \mathsf{Rec}, \mathsf{Wit})$ be a signature of arity n. We say that $(\mathsf{Pro}, \mathsf{Rec}, \mathsf{Wit})$ *represents* R (or $(\mathsf{Pro}, \mathsf{Rec}, \mathsf{Wit})$ is a signature of R) if the three following conditions are satisfied:

(C1) For every $k' \in [k-1]$, for every $I \subseteq [n]$ such that $|I| = k'$ and for every k'-ary tuple $\mathbf{a}$ on A, $(I, \mathbf{a}) \in \mathsf{Pro}$ if and only if $\mathbf{a} \in \mathrm{pr}_I\, R$.

(C2) For every $i \in [n]$ and for every minority pair $a, b \in A$, $(i, a, b) \in \mathsf{Rec}$ if and only if there are $\mathbf{a}, \mathbf{b} \in R$ such that $\mathrm{pr}_{[i-1]}\,\mathbf{a} = \mathrm{pr}_{[i-1]}\,\mathbf{b}$, $\mathrm{pr}_i\,\mathbf{a} = a$, and $\mathrm{pr}_i\,\mathbf{b} = b$.

(C3) Wit is contained in R.

Example 5 *Let us reconsider the operation f from Example 4. It is a trivial (although requiring substantial time) exercise to show that the relation*

$$R = \begin{pmatrix} c & c & c & c & c & c & b & a \\ a & a & b & b & a & b & c & c \\ a & b & a & b & c & c & c & c \end{pmatrix}$$

is invariant under f; the columns of the matrix represent the tuples from R. In order to construct a signature of this relation we, first, have to list all unary and binary projections of its tuples. Thus $\mathsf{Pro} = \{(\{1\}, (a)), (\{1\}, (b)), (\{1\}, (c)), (\{2\}, (a)), (\{2\}, (b)), (\{2\}, (c)), (\{3\}, (a)), (\{3\}, (b)), (\{3\}, (c)), (\{1,2\}, (c,a)), (\{1,2\}, (c,b)), (\{1,2\}, (a,c)), (\{1,2\}, (b,c)), (\{1,3\}, (c,a)), (\{1,3\}, (c,b)), (\{1,3\}, (a,c)), (\{1,3\}, (b,c)), (\{1,3\}, (c,c)), (\{2,3\}, (a,a)), (\{2,3\}, (a,b)), (\{2,3\}, (b,a)), (\{2,3\}, (b,b)), (\{2,3\}, (c,c)), (\{2,3\}, (a,c)), (\{2,3\}, (b,c))\}$.

Witnesses for the members of Pro *can easily be chosen. For example, (c,b,a) witnesses $(\{1,3\}, (c,a))$; note that it also witnesses $(\{1\}, (c))$, $(\{2\}, (b))$, $(\{3\}, (a))$, $(\{1,2\}, (c,b))$, and $(\{2,3\}, (b,a))$.*

Set Rec *is much smaller,* $\mathsf{Rec} = \{(2,a,b), (3,a,b)\}$. *Pairs of witnesses for them can be chosen to be $(c,a,a), (c,b,a)$ and $(c,b,a), (c,b,b)$, respectively.*

We prove that signatures are both polynomially evaluable and polynomially exactly learnable with equivalence queries. Thus the main results of this paper are the following two theorems.

Theorem 2 *Let R be an n-ary relation invariant under f. There exists an algorithm that, given a signature of R and a tuple $\mathbf{a} = (a_1, \ldots, a_n) \in A^n$, decides whether $\mathbf{a} \in R$ with running time polynomial in n.*

Theorem 3 *Let A be a finite set and let f be a GMM operation on A. Then, there exists an algorithm that exactly learns $\mathrm{Inv}(f)$, encoded with signatures, with equivalence queries and runs in time polynomial in the arity of the target relation.*

By Lemma 1, if Γ is a subset of $\mathrm{Inv}(f)$ for some operation f, then so is $\forall\exists$-Form(Γ). Therefore we deduce the following

Corollary 3 *Let A be a finite set, let f be a GMM operation on A and let Γ be a set of relations in $\mathrm{Inv}(f)$. Then $\forall\exists$-Form(Γ) is polynomially exactly learnable with (improper) equivalence queries.*

5 Proofs

In this section, we prove Theorems 2 and 3. Throughout the rest of the paper A will denote a finite set and f will be a GMM operation of arity k.

This section contains three subsections. In the first subsection we prove several simple properties of signatures. Then, in the second subsection we prove Theorem 2, and in the third subsection we present a learning algorithm for concept classes invariant under a GMM operation, thereby proving Theorem 3.

5.1 Properties of Signatures

We will need two simple properties of majority and minority pairs.

Lemma 2 *Let R be an n-ary relation invariant under f, and assume that $(\mathsf{Pro}, \mathsf{Rec}, \mathsf{Wit})$ is a signature that represents R.*

(1) Let $a_1, \ldots, a_n, b_1, \ldots, b_n$ be elements in A. If for every $j \in [n]$ the tuple $\mathbf{a}_j = (a_1, \ldots, a_{j-1}, b_j, a_{j+1}, \ldots, a_n)$ belongs to R and the pair a_n, b_n is majority, then $\mathbf{a} = (a_1, \ldots, a_n)$ belongs to R.

(2) (Rectangularity) If $a, b \in A$ constitute a minority pair, $(n, a, b) \in \mathsf{Rec}$, and $\mathbf{b} = (a_1, \ldots, a_{n-1}, b) \in R$, then $\mathbf{a} = (a_1, \ldots, a_{n-1}, a) \in R$.

Proof. (1) If for some j the pair a_j, b_j is minority, then $\mathbf{a} = f(\mathbf{a}_j, \ldots, \mathbf{a}_j, \mathbf{a}_n)$. If $\{a_j, b_j\}$ is majority for all j, then $\mathbf{a} = f(\mathbf{a}_1, \ldots, \mathbf{a}_{k-1}, \mathbf{a}_n)$.

(2) Since $(n, a, b) \in \mathsf{Rec}$, there exist elements $c_1, \ldots, c_{n-1} \in A$ such that $\mathbf{c} = (c_1, \ldots, c_{n-1}, a)$, $\mathbf{d} = (c_1, \ldots, c_{n-1}, b) \in R$. It is straightforward to check that

$$\mathbf{a} = f(\mathbf{b}, f(\mathbf{b}, \ldots, \mathbf{b}, \mathbf{d}), \ldots, f(\mathbf{b}, \ldots, \mathbf{b}, \mathbf{d}), f(\mathbf{b}, \ldots, \mathbf{b}, \mathbf{c})).$$

To check this, one can compute on every coordinate position, considering two cases depending on whether a_j, c_j is a minority or majority pair. □

Lemma 3 *Let R be an n-ary relation on A invariant under f and assume that $(\mathsf{Pro}, \mathsf{Rec}, \mathsf{Wit})$ is a signature of R. Then $\langle \mathsf{Wit} \rangle_f = R$.*

Proof. Let S be $\langle \mathsf{Wit} \rangle_f$. Since $\mathsf{Wit} \subseteq R$ we can conclude that $S \subseteq R$. We shall show that $R \subseteq S$. In particular, we shall show by induction on i that $\mathrm{pr}_{[i]} R \subseteq \mathrm{pr}_{[i]} S$. Take $i \leq n$. If $i \leq k-1$ then the required inclusion easily follows from condition (C1) of the definition of signature. So let $i \geq k$, $R' = \mathrm{pr}_{[i]} R$, $S' = \mathrm{pr}_{[i]} S$, and $\mathbf{a} = (a_1, \ldots, a_i) \in R'$. By the induction hypothesis, for some b_i, the tuple $(a_1, \ldots, a_{i-1}, b_i) = \mathbf{a}'$ belongs to S'.

We consider two cases.

Case 1. $\{a_i, b_i\}$ is majority.

In this case we show that, for every $I \subseteq [i-1]$, $\mathrm{pr}_{I \cup \{i\}} \mathbf{a} \in \mathrm{pr}_{I \cup \{i\}} S'$. We show it by induction on the cardinality m of I. The result is true for $m \leq k-2$ again by (C1). Thus let $I = \{j_1, \ldots, j_m\}$ be any set of indices $1 \leq j_1 < j_2 < \cdots < j_m < i$ with $m \geq k$. By induction hypothesis, for any $\ell \in [m]$, there is b_ℓ such that $(a_{j_1}, \ldots, a_{j_{\ell-1}}, b_\ell, a_{j_{\ell+1}}, \ldots, a_{j_m}) \in \mathrm{pr}_{I \cup \{i\}} S'$. Recall also that $\mathrm{pr}_{I \cup \{i\}} \mathbf{a}' = (a_{j_1}, \ldots, a_{j_m}, b_i)$ also belongs to $\mathrm{pr}_{I \cup \{i\}} S'$. By Lemma 2(1), $\mathrm{pr}_{I \cup \{i\}} \mathbf{a} = (a_{j_1}, \ldots, a_{j_m}, a_i) \in \mathrm{pr}_{I \cup \{i\}} S'$.

Case 2. $\{a_i, b_i\}$ is minority.

Since $\mathbf{a}' \in S'$ and $S \subseteq R$, we have $\mathbf{a}' \in R'$. Furthermore, since $\mathbf{a} \in R'$, we can conclude that (i, a_i, b_i) belongs to Rec. By Lemma 2(2), $\mathbf{a} \in S'$. □

Corollary 4 *Let R and S be n-ary relations on A invariant under f with signatures $(\mathsf{Pro}_R, \mathsf{Rec}_R, \mathsf{Wit}_R)$ and $(\mathsf{Pro}_S, \mathsf{Rec}_S, \mathsf{Wit}_S)$. If the signatures are identical then $R = S$.*

From Corollary 4, it is clear that a signature represents at most one relation invariant under f. However, it is possible that a given signature does not represent any relation invariant under f. A simple example is as follows. Consider the GMM operation f defined by $f(x, y, z) = x - y + z$ over Z_3. Let $(\mathsf{Pro}, \mathsf{Rec}, \mathsf{Wit})$ be a signature of arity one with $\mathsf{Pro} = \{(\{1\}, (0)), (\{1\}, (1))\}$, $\mathsf{Rec} = \{(1, 0, 1)\}$, and $\mathsf{Wit} = \{(0), (1)\}$. Any relation R containing Wit and closed under f must also contain $f(0, 1, 0) = 2$, that is, $R = \{(0), (1), (2)\}$. But, a signature for such a relation R must have $(\{1\}, 2) \in \mathsf{Pro}$ and it must also have $(1, 0, 2), (1, 1, 2) \in \mathsf{Rec}$.

Before embarking on the study of signatures we shall derive some easy facts from Theorem 3 which allow us to shed some light on the closure algorithm. Let R be any arbitrary relation in $\mathrm{Inv}(f)$ and let $\mathbf{x_1}, \ldots, \mathbf{x_m}$ be the sequence of counterexamples provided to the closure algorithm when it is executed with target concept R. For every $i \in \{0, \ldots, m\}$ let us denote by R_i the relation generated by $\mathbf{x_1}, \ldots, \mathbf{x_i}$. Fix, for each $i \in \{1, \ldots, m\}$, a signature $(\mathsf{Pro}_i, \mathsf{Rec}_i, \mathsf{Wit}_i)$ of R_i. Since, R_{i-1} is a subset of R_i then we must have $\mathsf{Pro}_{i-1} \subseteq \mathsf{Pro}_i$ and $\mathsf{Rec}_{i-1} \subseteq \mathsf{Rec}_i$. Furthermore, $\mathsf{Pro}_{i-1} \neq \mathsf{Pro}_i$ or $\mathsf{Rec}_{i-1} \neq \mathsf{Rec}_i$ since otherwise $(\mathsf{Pro}_{i-1}, \mathsf{Rec}_{i-1}, \mathsf{Wit}_{i-1})$ would be a signature of R_i in contradiction with the fact that $R_{i-1} \neq R_i$. Consequently, at each iteration i of the closure algorithm, the size of Pro_i or Rec_i increases. Hence the total number of iterations is bounded by the value of $|\mathsf{Pro}| + |\mathsf{Rec}|$ where $(\mathsf{Pro}, \mathsf{Rec}, \mathsf{Wit})$ is a signature of R.

Notice that, as the number of at most $(k-1)$-element subsets of an n-element set is bounded by n^{k-1}, and the number of $(k-1)$-tuples of elements of A is $|A|^{k-1}$, the size of Pro is bounded by $n^{k-1} \cdot |A|^{k-1}$. Similarly, the maximal number of elements that Rec may contain is $n \cdot |A|^2$. Since f and hence k and A are fixed, the size of Pro and Rec (and hence of Wit) is always bounded by a polynomial in n. We state this as a lemma for future use.

Lemma 4 *Let f be a GMM operation of arity k. The is a constant α such that, for any signature $(\mathsf{Pro}, \mathsf{Rec}, \mathsf{Wit})$ with respect to f of arity n, we have $|\mathsf{Pro}|, |\mathsf{Rec}|, |\mathsf{Wit}| \leq \alpha n^{k-1}$.*

This line of reasoning shows that the closure algorithm always stops in a polynomial number of steps and that every relation in $\mathrm{Inv}(f)$ has a generating set with size polynomial in n. This would be the end of the story *if* generating sets were polynomially evaluable. However, we do not know how to decide in polynomial time whether a given tuple belongs to the relation generated by its generating set.

To overcome this difficulty, we investigate the learnability of signatures.

We have to find an algorithm that decides in polynomial time, given a tuple $\mathbf{a}$ and a signature s, whether the tuple $\mathbf{a}$ belongs to the relation represented by the signature s.

Let R be an n-ary relation invariant under f, let $s = (\mathsf{Pro}, \mathsf{Rec}, \mathsf{Wit})$ be a signature of R, and let $a_1, \ldots, a_m \in A$, $m \leq n$. By $R_{a_1,\ldots,a_m}$ we denote the relation

$$\{(b_1, \ldots, b_n) \in R \mid \mathrm{pr}_{[m]}(b_1, \ldots, b_n) = (a_1, \ldots, a_m)\}$$

A signature $s_{a_1,\ldots,a_m}$ of $R_{a_1,\ldots,a_m}$ can be efficiently computed as the following lemma shows.

Lemma 5 *Let R be an n-ary relation invariant under f and $a_1, \ldots, a_m \in A$. There exists an algorithm that, given a signature $s = (\mathsf{Pro}, \mathsf{Rec}, \mathsf{Wit})$ of R computes a signature $s_{a_1,\ldots,a_m}$ of $R_{a_1,\ldots,a_m}$ with running time polynomial in n.*

Proof. Let d be any element in the universe A. We shall show an algorithm that constructs a signature $s_d = (\mathsf{Pro}_d, \mathsf{Rec}_d, \mathsf{Wit}_d)$ of R_d. First, we note two properties of signature s_d.

(1) For any $I \subseteq [n]$ with cardinality $k' < k$, and any $\mathbf{a} \in A^{k'}$, we have $(I, \mathbf{a}) \in \mathsf{Pro}_d$ if and only if there exists some $\mathbf{b}$ in $\mathrm{pr}_{I \cup \{1\}} R$ such that $\mathrm{pr}_I \mathbf{b} = \mathbf{a}$ and $\mathrm{pr}_1 \mathbf{b} = d$.

(2) For any $i \in [n]$ and a minority pair $\{a, b\}$, we have $(i, a, b) \in \mathsf{Rec}_d$ if and only if $(i, a, b) \in \mathsf{Rec}$ and $(\{i\}, (a)) \in \mathsf{Pro}_d$.

Our algorithm uses properties (1) and (2) to produce s_d. By (1), to compute Pro_d and the elements of Wit_d witnessing the tuples from Pro_d, it suffices to compute $S = \langle \mathrm{pr}_J \mathsf{Wit} \rangle_f$ for every $J = I \cup \{1\}$, for all subsets I of $[n]$ with cardinality $k' < k$, and choose from S those tuples which have d as the first component. Since S is a relation of arity at most k (fixed) and the maximum possible total number of tuples in S is $|A|^k$, S can be computed efficiently by a brute-force algorithm. This algorithm applies f, firstly, to tuples from $\mathrm{pr}_J \mathsf{Wit}$ and then to newly generated tuples, and keeps a record on the sequence of applications of operation f that ends up with some tuple $\mathbf{a}$. Such a record can later be used to produce a tuple $\mathbf{b} \in R$ with $\mathrm{pr}_J \mathbf{b} = \mathbf{a}$, which witnesses $(I, \mathrm{pr}_I \mathbf{a})$ in Wit_d.

Property (2) is used to generate Rec_d. Since Pro_d is already known, for any triple $(i, a, b) \in \mathsf{Rec}$, it is easy to check whether or not $(\{i\}, (a))$ belongs to Pro_d. Moreover, if $(\{i\}, (a)) \in \mathsf{Pro}_d$ then Wit_d contains a tuple $\mathbf{b}$ such that $\mathrm{pr}_1 \mathbf{b} = d$ and $\mathrm{pr}_i \mathbf{b} = a$; also, since $(i, a, b) \in \mathsf{Rec}$ there are $\mathbf{c}, \mathbf{d} \in \mathsf{Wit}$ with $\mathrm{pr}_{[i-1]} \mathbf{c} = \mathrm{pr}_{[i-1]} \mathbf{d}$ and $\mathrm{pr}_i \mathbf{c} = a, \mathrm{pr}_i \mathbf{d} = b$. Applying the same trick

as in the proof of Lemma 2(2) we obtain $\mathbf{a}$ such that $\mathrm{pr}_{[i-1]}\,\mathbf{a} = \mathrm{pr}_{[i-1]}\,\mathbf{b}$, $\mathrm{pr}_1\,\mathbf{b} = \mathrm{pr}_1\,\mathbf{b} = d$, and $\mathrm{pr}_i\,\mathbf{a} = b$. Thus $\mathbf{b}, \mathbf{a}$ can be included into Wit_d to witness (i, a, b).

Finally, we prove properties (1) and (2). First, $(I, \mathbf{a}) \in \mathsf{Pro}_d$, $\mathbf{a} = (a_1, \ldots, a_{k'})$, if and only if there is a tuple $\mathbf{c} \in R$ such that $\mathrm{pr}_I\,\mathbf{c} = \mathbf{a}$ and $\mathrm{pr}_1\,\mathbf{c} = d$. We only need to set $\mathbf{b}$ to $\mathrm{pr}_J\,\mathbf{c}$ to complete the proof.

To prove (2) we observe that if $(i, a, b) \notin \mathsf{Rec}$ or $(\{i\}, (a)) \notin \mathsf{Pro}_d$ then (i, a, b) is not in Rec_d, since no pair of tuples from R_d witnesses it. Otherwise such a pair exists as shown above.

By iterating the process we can obtain, for any given $a_1, \ldots, a_l \in A$, a signature of $R_{a_1,\ldots,a_l} = (\ldots(R_{a_1})_{a_2})_{\ldots})_{a_l}$. It is easy to see that the running time of the algorithm is polynomial in n. $\square$

Now we are ready to prove Theorem 2

Proof. To check if a given tuple $(a_1, \ldots, a_n)$ belongs to R, it suffices to check whether the set $\mathsf{Wit}(R_{a_1,\ldots,a_n})$ is non-empty. We assume by convention that there exists one tuple of arity 0. $\square$

We have seen above that there are signatures that do not necessarily represent any relation in $\mathrm{Inv}(f)$. We are interested in associating to each of them a relation (which is not necessarily in $\mathrm{Inv}(f)$). The reason for this is that our learning algorithm might produce, as an intermediate hypothesis, some signature that does not necessarily represent a relation. In order to deal with this issue we define the concept *encoded* by a signature $(\mathsf{Pro}, \mathsf{Rec}, \mathsf{Wit})$ as the relation containing all those tuples $\mathbf{a}$ accepted by the evaluation algorithm. In the case that a signature $(\mathsf{Pro}, \mathsf{Rec}, \mathsf{Wit})$ indeed represents a relation in $\mathrm{Inv}(f)$, the concept encoded by $(\mathsf{Pro}, \mathsf{Rec}, \mathsf{Wit})$ and the concept represented by $(\mathsf{Pro}, \mathsf{Rec}, \mathsf{Wit})$ coincide.

Notice that if we run the algorithm over signatures that do not represent a relation invariant under f then the output of the algorithm cannot be completely predicted. However, by a mere inspection of the proof one thing can be said: when given as input a tuple $\mathbf{a}$ and a signature $(\mathsf{Pro}, \mathsf{Rec}, \mathsf{Wit})$, if the evaluation algorithm outputs "yes", then $\mathbf{a}$ belongs to $\langle \mathsf{Wit} \rangle_f$. Let us state this as a fact for future reference.

Fact 1 *The concept encoded by a signature* $(\mathsf{Pro}, \mathsf{Rec}, \mathsf{Wit})$ *is always a subset of* $\langle \mathsf{Wit} \rangle_f$.

An algorithm that learns signatures with equivalence queries is presented in Figure 1.

The target concept of the algorithm is an n-ary relation invariant under f. Recall that f, k and A are fixed. At any stage of the execution of the algorithm Wit contains only elements of R. Hence, by Fact 1, the relation represented by $(\mathsf{Pro}, \mathsf{Rec}, \mathsf{Wit})$ is a subset of $\langle \mathsf{Wit} \rangle_f \subseteq R$. This ensures that the algorithm is monotonic, that is, only receives positive counterexamples. Furthermore at every round of the algorithm, either Pro or Rec increases its size. Since Pro and Rec can have at most a polynomial number of elements we can ensure that the algorithm ends in a number of steps polynomial in n.

Step 1. **set** $\mathsf{Pro}, \mathsf{Rec}, \mathsf{Wit} := \emptyset$
Step 2. **while** $EQ((\mathsf{Pro}, \mathsf{Rec}, \mathsf{Wit}))$= 'no' **do**
  let $\mathbf{a} = (a_1, \ldots, a_n)$ be the counterexample produced by EQ
Step 2.1 **if** for some $I = \{i_1, \ldots, i_{k'-1}\}$, $i_1 < i_2 < \cdots < i_{k'-1}$, $k' \leq k$,
    the pair $(I, \mathrm{pr}_I \mathbf{a}) \notin \mathsf{Pro}$ **then**
Step 2.1.1 **set** $\mathsf{Pro} := \mathsf{Pro} \cup (I, \mathrm{pr}_I \mathbf{a})$ and $\mathsf{Wit} := \mathsf{Wit} \cup \{\mathbf{a}\}$
Step 2.2 **else**
    For each $0 \leq i \leq n$ compute $(\mathsf{Pro}_{a_1,\ldots,a_i}, \mathsf{Rec}_{a_1,\ldots,a_i}, \mathsf{Wit}_{a_1,\ldots,a_i})$.
    Let i be the smallest integer such that $\mathsf{Wit}_{a_1,\ldots,a_i} = \emptyset$
      (such i must exist if $\mathbf{a}$ does not belong
      to the relation represented by $(\mathsf{Pro}, \mathsf{Rec}, \mathsf{Wit})$)
Step 2.2.1 **pick** an element $\mathbf{b} = (a_1, \ldots, a_{i-1}, b_i, \ldots, b_n) \in \mathsf{Wit}_{a_1,\ldots,a_{i-1}}$
Step 2.2.2 **set** $\mathsf{Rec} := \mathsf{Rec} \cup \{(i, a_i, b_i)\}$ and $\mathsf{Wit} := \mathsf{Wit} \cup \{\mathbf{a}, \mathbf{b}\}$
  endwhile
Step 3. **return** $\mathsf{Pro}, \mathsf{Rec}, \mathsf{Wit}$

Fig. 1. Algorithm learning signatures with equivalence queries. EQ indicates an equivalence query.

Let us study correctness of this algorithm. The algorithm increases Pro whenever possible (Step 2.1.1). Otherwise some triple has to be added to Rec (Step 2.2). There is only one thing that requires some proof: we have to show that at Step 2.2.2, Rec is increased correctly, that is, it is the case that a_i, b_i is a minority pair and that (i, a_i, b_i) is not previously in Rec.

Suppose that a_i, b_i is a majority pair. We prove by induction that $\mathrm{pr}_{J \cup \{i\}} \mathbf{a} \in \mathrm{pr}_{J \cup \{i\}} \langle \mathsf{Wit} \rangle_f$ for any $J \subseteq [i-1]$ obtaining a contradiction with $\mathsf{Wit}_{a_1,\ldots,a_i} = \emptyset$. For the base case of induction we note that this inclusion holds for any J with $|J| < k$, because otherwise Step 2.1.1 would be performed. Suppose that the inclusion is true for any J with $|J| < \ell$, and take $J = \{i_1, \ldots, i_\ell\}$ where $1 \leq i_1 < \ldots < i_\ell < i$. Denoting $J_m = J \setminus \{i_m\}$, by induction hypothesis, $\mathrm{pr}_{J_m \cup \{i\}} \mathbf{a} \in \mathrm{pr}_{J_m \cup \{i\}} \langle \mathsf{Wit} \rangle_f$ for any $m \leq \ell$. This means that there

are $b_1, \ldots, b_\ell$ such that the tuples $\mathbf{a}_m = (a_{i_1}, \ldots, a_{i_{m-1}}, b_m, a_{i_{m+1}}, \ldots, a_{i_\ell}, a_i)$ belong to $\mathrm{pr}_{J\cup\{i\}}\langle \mathsf{Wit}\rangle_f$. By Lemma 2(1), $\mathrm{pr}_{J\cup\{i\}}\,\mathbf{a} \in \mathrm{pr}_{J\cup\{i\}}\langle \mathsf{Wit}\rangle_f$, a contradiction.

Next we show that $(i, a_i, b_i) \notin \mathsf{Rec}$. Let us suppose for contradiction that $(i, a_i, b_i) \in \mathsf{Rec}$. Then for some elements $c_1, \ldots c_{i-1}$ we have that the tuples $(c_1, \ldots c_{i-1}, a_i)$ and $(c_1, \ldots c_{i-1}, b_i)$ are in $\mathrm{pr}_{[i]}\langle \mathsf{Wit}\rangle_f$. By Lemma 2(2), as $(a_1, \ldots a_{i-1}, b_i) \in \mathrm{pr}_{[i]}\langle \mathsf{Wit}\rangle_f$, we have $(a_1, \ldots a_{i-1}, a_i) \in \mathrm{pr}_{[i]}\langle \mathsf{Wit}\rangle_f$, a contradiction.

This completes the proof of the correctness of the algorithm.

We have just proved that there exists an algorithm that exactly learns $\mathrm{Inv}(f)$, encoded with signatures, with equivalence queries in polynomial time.

Finally, Theorem 3 is obtained by combining this result with the fact that the size of a signature is bounded by a polynomial in the arity of the relation it encodes (Lemma 4).

Acknowledgements. The first author was supported by an NSERC Discovery Grant. The second and third authors were supported by grant TIC 2002-04470-C03 and the EU PASCAL Network of Excellence IST-2002-506778. The third author was also supported by the MEC under the program "Ramon y Cajal", grant TIC 2002-04019-C03, and MODNET Marie Curie Research Training Network MRTN-CT-2004-512234.

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] P. Auer and N. Cesa-Bianchi. On-Line Learning with Malicious Noise and the Closure Algorithm Ann. Math. Artif. Intell., 23(1-2):83-99.

[5] F. Börner, A.A. Bulatov, P.G. Jeavons, and A.A. Krokhin. Quantified constraints and surjective polymorphisms. In *Proceedings of 17th International Workshop Computer Science Logic, CSL'03*, volume 2803 of *Lecture Notes in Computer Science*, pages 58–70. Springer-Verlag, 2003.

[6] N. Bshouty, J. Jackson, and C. Tamon. Exploiring Learnability between Exact and PAC. In *15th Annual ACM Conference on Computational Learning Theory, COLT'02*, pages 244–254, 2002.

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

[8] A.A. Bulatov. Tractable conservative constraint satisfaction problems. In *Proceedings of the 18th Annual IEEE Simposium on Logic in Computer Science*, pages 321–330, Ottawa, Canada, June 2003. IEEE Computer Society.

[9] H. Chen. The Computational Complexity of Quantified Constraint Satisfaction. PhD Thesis, Cornell University, 2004.

[10] H. Chen and V. Dalmau. From Pebble Games to Tractability: An Ambidextrous Consistency Algorithm for Quantified Constraint Satisfaction. Computer Science Logic (CSL), 2005.

[11] H Chen. Quantified Constraint Satisfaction, Maximal Constraint Languages, and Symmetric Polymorphisms. 22nd International Symposium on Theoretical Aspects of Computer Science (STACS), 2005.

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

[13] N. Creignou and M. Hermann. Complexity of Generalized Satisfiability Counting Problems. *Information and Computation*, 125:1–12, 1996.

[14] N. Creignou, S. Khanna, and M. Sudan. *Complexity Classification of Boolean Constraint Satisfaction Problems*, volume 7 of *Monographs on Discrete Mathematics and Applications*. SIAM, 2001.

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

[16] V. Dalmau and P. Jeavons. Learnability of Quantified Formulas. *Theoretical Computer Science*, (306):485–511, 2003.

[17] K. Denecke, M. Erne, and S.L. Wismath. Galois Connections and Applications. Kluwer Academic Publishers, 2003.

[18] D. Haussler, N. Littlestone, and M. Warmuth. Predicting $\{0,1\}$-functions on randomly drawn points. *Information and Computation*, 115(2):248–292.

[19] D. Hembold, R. Sloan, and M. Warmuth. Learning Nested Differences of Intersection-Closed Concept Classes . *Machine Learning*, 5:165–196.

[20] D. Hembold, R. Sloan, and M. Warmuth. Learning Integer Lattices . *SIAM J. Computing*, 21(2):240-266.

[21] 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.

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

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

[24] S. Khanna, M. Sudan, and L. Trevisan. Constraint Satisfaction: The Approximability of Minimization Problems. In *12th IEEE Conference on Computational Complexity*, 1997.

[25] S. Khanna, M. Sudan, and P.D. Williamson. A Complete Classification of the Approximability of Maximation Problems Derived from Boolean Constraint Satisfaction. In *29th Annual ACM Symposium on Theory of Computing*, 1997.

[26] N. Littlestone. Learning Quickly when Irrelevant Atributes Abound: A New Linear Threshold Algorithm. *Machine Learning*

[27] B. K. Natarajan. On learning Boolean functions. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, 1987.

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

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

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

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

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