Computational Complexity of Problems over Generalized Formulas

Víctor Dalmau

dalmau@lsi.upc.es

Thesis presented to obtain the Phd. degree in the Departament de Llenguatges i Sistemes Informàtics (LSI) at the Universitat Politécnica de Catalunya (UPC)

Advisor: Ricard Gavaldà

March 6, 2000

Abstract

In this thesis we define a wide class of formulas, called *generalized quantified formulas* which are versatile and expressive enough to contain many natural families of logical formulas and representation schemes, such as constraint networks. A generalized quantified formula has the following form: (1) fix a set of basic relations called a basis, (2) take conjunctions of these basic relations applied to constants and variables in arbitrary ways, and (3) finally, quantify existentially or universally some of the variables.

The main goal of the thesis is to study the computational complexity of solving some problems over generalized quantified formulas. We focus mainly on two computational problems: *satisfiability* and *learnability*.

We start by studying generalized quantified Boolean formulas, i.e., generalized quantified formulas over a basis containing Boolean relations. We prove a sharp dichotomy for the learnability problem. More precisely, we show that for any set of basic Boolean relations, the resulting set of formulas is either polynomially learnable from equivalence queries alone or else it is not PAC-predictable even with membership queries under cryptographic assumptions. Furthermore we identify precisely which sets of basic functions are in which of the two cases.

We also study the satisfiability of quantified Boolean formulas. Schaefer earlier studied the satisfiability of generalized quantified Boolean formulas with constants. He showed that every such problem is either in P or PSPACE-complete and he gave a complete classification of the tractable cases. We extend the PSPACE-hardness results to constant-free quantified Boolean formulas obtaining a dichotomy theorem for the satisfiability of constant-free quantified Boolean formulas.

Then, we study the general case, i.e., formulas over not necessarily Boolean basis. We introduce as an intermediate tool, a link between generalized quantified formulas and some well-studied structures in Universal Algebra called

clones.

We study first the satisfiability problem over non-quantified generalized formulas, also known as constraint satisfaction problem. In this setting, we identify two new families of problems solvable in polynomial time: the class of set function problems and the class of para-primal problems. The class of set function problems is interesting because it corresponds to the class of problems that can be solved using arc-consistency. As an application of the tractability of para-primal problems and some results about the structure of homogeneous algebras, we obtain a complete classification for constraint satisfaction problems under the assumption that the basis contains all the permutation relations.

We also extend the learnability results to the general case. We identify three efficiently learnable families of generalized quantified formulas containing, as a particular case, the learnable cases in the Boolean domain. Furthermore, we show that, under some restrictions in the structure of the formulas, the list of learnable cases is complete.

Finally, we use the link between formulas and clones and the fact that the structure of clones is well known in the Boolean case to study other computational problems over Boolean formulas. More precisely, we study the inverse satisfiability problem, also known as structure identification problem, over generalized quantified formulas and we study the complexity of learning Boolean circuits for a fixed basis of gates obtaining a complete classification under some cryptographic assumptions.

Acknowledgments

I want to express my most deep gratitude to my advisor Ricard Gavaldà for his continuous encouraging and support, for his willingness to discuss any question not merely concerning research, and for his infinite patience to deal with my innate sense of procrastination.

I am also greatly indebted to Peter Jeavons and Dave Cohen, for introducing me to the non-binary universe, and for teaching me all what I know about constraint satisfaction and clones. Their influence is spread along all the pages of this thesis.

I want to thank Justin Pearson for inviting me to Uppsala, for the pleasure of collaborate with him, and for numerous comments on this thesis and its presentation.

I want to thank to my former officemates at LSI: Albert Atserias, Jorge Castro, Carlos Domingo, David Guijarro and Víctor Lavín for their help and collaboration.

Finally, I specially thank to my parents their support.

This work has been partly supported by: the Spanish Government through grant AP94 43716784 and 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 8556 (NeuroCOLT), Working Group 27150 (NeuroCOLT2), project 20244 (ALCOM IT) and project TIC97-1475-CE (CICYT).

Contents

List of Figures

Chapter 1

Introduction

A large number of computational problems encountered in complexity theory, such as the satisfiability problem, can be classified as the study of computational properties of certain kinds of logical formulas. Furthermore, logical formulas are a formalism that is widely used in many applications, such as knowledge-representation in artificial intelligence, reasoning, databases and logic circuit design. In consequence, the study of computational properties over logical formulas is an important problem that arises both in theory and practice.

The inevitable trade-off between the expressive power of a family of formulas and the resources needed to solve any computational property over them has forced researchers to study restricted classes of formulas.

Example 1 Consider the problem of deciding whether a logical formula has a solution, also called the *satisfiability problem* (SAT). The satisfiability problem is known to be NP-complete [13] for formulas in conjunctive normal form (CNF) and several solvable subclasses have been identified.

Such subclasses can be obtained by restricting the ways in which variables may be constrained with other variables, as for instance, read-2-CNF [71], or they can be obtained by restricting the clauses allowed to built the formula, as for instance, Horn CNF [29] or 2-CNF [13].

Schaefer [81] proposed a unifying framework to study the complexity of satisfiability problems obtained restricting the set of clauses, or even more generally, the relations used to construct a formula. In Schaefer's model, a family of formulas is characterized by basis or finite set $\Gamma = \{R_1, \ldots, R_m\}$ of boolean relations where Γ captures the notion of the different ways in which variables can be "constrained". For every basis Γ, Schaefer defined an associated class of formulas called *generalized formulas over* Γ, denoted by Form(Γ). Every formula in Form(Γ) is the conjunction of some relations in Γ applied to variables in any arbitrary way (for formal definition see Section 2.1.1).

Example 2 Consider the problem of deciding the satisfiability of a formula formed by a conjunction of clauses with three literals per clause. Every such formula can be expressed as a formula in Form(Γ) with the set of logical relations $\Gamma = \{R_0, R_1, R_2, R_3\}$, defined by:

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

Schaefer proved that for every finite boolean basis Γ, the satisfiability of Form(Γ) is either solvable in polynomial time or NP-complete. Furthermore, he gave a complete classification of the tractable problems. He showed that the satisfiability of Form(Γ) is polynomial-time decidable if and only if at least one of the following conditions holds:

(a) Every relation in Γ is definable by a CNF formula in which each clause has at most 2 literals (Γ is bijunctive).

(b) Every relation in Γ is the set of solutions of a system of linear equations over the two-element field $\{0, 1\}$ (Γ is affine).

(c) Every relation in Γ is definable by a CNF formula in which each clause has at most one affirmed variable (Γ is horn).

(d) Every relation in Γ is definable by a CNF formula in which each clause has at most one negated variable (Γ is antihorn).

(e) Every non-void relation in Γ is satisfied by the assignment $(0, \ldots, 0)$ (Γ is 0-valid).

(f) Every non-void relation in Γ is satisfied by the assignment $(1, \ldots, 1)$ (Γ is 1-valid).

To our knowledge, Schaefer's result remained largely unnoticed for a good number of years. Recently, Schaefer's dichotomy has inspired several results on other problems over generalized formulas. The ones known to us concern the inverse generalized satisfiability problem [54], the generalized satisfiability counting problem [17], the complexity of computing optimal assignments [78], and the approximability of minimization and maximization problems [16, 56, 57].

The main goal of this thesis is to further continue this line of research by studying some computational properties over some extensions of generalized formulas. Our extension is obtained by endowing generalized formulas, as defined by Schaefer, with two new natural capabilities: (1) quantification and (2) non-boolean domains.

Given a finite basis $\Gamma = \{R_1, \ldots, R_m\}$, where every R_i is a relation over an arbitrary domain D. The class of generalized quantified formulas over Γ, denoted by $\exists\forall\text{-}\operatorname{Form}(\Gamma)$, is set of formulas obtained as the conjunction of some relations in Γ applied to variables in any arbitrary way, and finally, with some of the variables quantified existentially or universally in any arbitrary way. We will be also interested in some classes of generalized formulas obtained restricting the previous definition in several ways, such as allowing only existential quantification or no quantification at all or, conversely, extending the previous definition allowing, for example, to apply constants in D to relations. We will refer as generalized formulas to the whole family of formulas, including the generalized quantified formulas and all the related families obtained from them by restricting or extending in any of the previous ways.

Our long-term project is to study different computational properties over generalized formulas. The previous results about generalized boolean formulas fit perfectly in this scheme. In this thesis we focus on two computational problems of particular interest in artificial intelligence: satisfiability problems and learnability problems.

We have discussed satisfiability problems previously. Satisfiability problems over arbitrary (non-necessarily boolean) domains are generally known as constraint satisfaction problems.

The problem of learning an unknown boolean formula under some determined protocol has been widely studied in computer learning theory. It is well known that, even restricted to propositional formulas, the problem is hard [3, 55] in the usual formal models. Therefore researchers have attempted to learn subclasses of propositional boolean formulas, specially inside CNF and DNF. For example, k-CNF formulas, k-term CNF formulas, monotone-CNF formulas, Horn formulas, and their dual counterparts [1, 6, 2] have all been shown exactly learnable using membership and equivalence queries in Angluin's model [1] while the question of whether CNF formulas are learnable is still open. The more powerful formalism of predicate logics is used in several applications of learning in artificial intelligence and knowledge representation but its study, from the computational learning theory point of view, is recent. See [60] and the further references in that paper. An active line of research in predicate logics is, for instance, *Inductive Logic Programming (ILP)* [67, 68, 58].

The methodology used to study problems over the boolean domain and larger domains are completely different. Thus, we present the results separately.

1.1 Boolean Case

1.1.1 Satisfiability Problems

The study of the complexity of deciding the satisfiability of generalized formulas over a boolean basis was almost completed by Schaefer [81]. We stated above Schaefer's dichotomy theorem for the satisfiability of generalized (quantifier-free) formulas over a boolean basis.

Schaefer also studied the satisfiability of generalized quantified boolean formulas obtaining a dichotomic classification for a slightly different family of formulas called *generalized quantified formulas with constants over a boolean basis* Γ, denoted by $\exists\forall\text{-}\,\mathrm{Form}_C(\Gamma)$. Schaefer proved that for every finite boolean Γ, the satisfiability of $\exists\forall\text{-}\,\mathrm{Form}_C(\Gamma)$ is polynomially decidable if and only if at least one of the following conditions holds:

(a) The basis Γ is bijunctive.

(b) The basis Γ is affine.

(c) The basis Γ is horn.

(d) The basis Γ is antihorn.

Otherwise, the problem is PSPACE-complete.

In Chapter 4 we complete the picture on satisfiability of generalized formulas by proving the dichotomy theorem for generalized (constant-free) quantified formulas over a finite boolean basis. Our classification is identical to Schaefer's dichotomy for generalized quantified formulas with constants. We state it for completeness: the satisfiability of $\exists\forall$- $\mathrm{Form}(\Gamma)$ (for finite Γ) is polynomially decidable if and only if at least one of the following conditions holds:

(a) The basis Γ is bijunctive.

(b) The basis Γ is affine.

(c) The basis Γ is horn.

(d) The basis Γ is antihorn.

Otherwise, the problem is PSPACE-complete.

A remarkable point of our result is that it is obtained as a corollary of a more general result that characterizes the cases in which the use of constants increases the expressiveness power of a class. We prove that, in a sense that will be precised later, $\exists\forall$- $\mathrm{Form}_C(\Gamma)$ is more expressive than $\exists\forall$- $\mathrm{Form}(\Gamma)$ if and only if at least one of the following conditions holds:

(a) Every relation in Γ is definable by a conjunction of clauses in which each clause is either a monotone disjunction of literals or the equality between two variables (Γ is equality monotone).

(b) Every relation in Γ is definable by a conjunction of clauses in which each clause is either a antimonotone disjunction of literals or the equality between two variables (Γ is equality antimonotone).

(c) Every relation in Γ is the set of solutions of an homogeneous system of linear equations over the two-element field $\{0, 1\}$ (Γ is 0-affine).

(d) Every relation in Γ is the set of solutions of a system of a linear equations over the two-element field $\{0, 1\}$ with particular solution $\langle 1, \ldots, 1\rangle$ (Γ is 1-affine).

This result is stated purely in terms of expressiveness power, without involving complexity notions, in order to be applicable to different contexts. We make use of it studying the learning complexity of quantified boolean formulas.

1.1.2 Learnability Problems

The problem of learning (quantifier-free) generalized boolean formulas is easy even in the simplest learning models (for finite basis). In consequence, we focus on the learnability of generalized quantified formulas over a boolean basis. In Chapter 3, we show that for every finite boolean basis Γ, the class of generalized quantified formulas over Γ ($\exists\forall$- Form(Γ)) is either polynomially learnable in the simple model of exact learning with equivalence queries [1] or not polynomially predictable even in the powerful model of PAC-prediction with membership queries [3] under some cryptographic assumptions. Furthermore, we give a complete classification of the tractable classes. More precisely we show that, in the sense precised above, $\exists\forall$- Form(Γ) is learnable if and only if the basis Γ satisfies at least one of the following conditions:

(a) The basis Γ is bijunctive.

(b) The basis Γ is affine.

(c) Every relation in Γ is definable by a CNF formula in which each clause has no affirmed variables or has at most one affirmed variable and at most one negated variable (Γ is weakly antimonotone).

(d) Every relation in Γ is definable by a CNF formula in which each clause has no negated variables or has at most one negated variable and at most one affirmed variable (Γ is weakly monotone).

As an intermediate step in the proof of the previous result, we use a similar dichotomic classification of the learning complexity of generalized quantified boolean formulas with constants. It is remarkable that, similarly to the satisfiability problem, the classification of learnable classes remain unchanged regardless of whether constants are available.

Figure 1.1 summarizes the positive cases for the satisfiability and learnability of generalized boolean formulas.

1.2 Larger Domains

In the study of computational problems over larger domains the "elementary methods" used in the boolean case are not valid any more. To study larger domains we consider a completely different approach based in some algebraic

	Non-Quantified		Quantified	
	$\mathrm{Form}_C(\Gamma)$ (with const.)	$\mathrm{Form}(\Gamma)$ (const.-free)	$\exists\forall$- $\mathrm{Form}_C(\Gamma)$ (with const.)	$\exists\forall$- $\mathrm{Form}(\Gamma)$ (const.-free)
Satisf.	bijunctive affine horn antihorn	bijunctive affine horn antihorn 0-valid 1-valid	bijunctive affine horn antihorn	bijunctive affine horn antihorn
Learn.	All cases	All cases	bijunctive affine weakly antimon. weakly mon.	bijunctive affine weakly antimon. weakly mon.

Figure 1.1: Summary of results in the boolean case

properties of the bases. The key notion of our approach is that of *closure operation* introduced by Jeavons et al. [51] in the study of constraint satisfaction problems (see Section 5.2 for a formal definition of closure operation).

In Chapter 6 we revisit the connection between closure operations and satisfiability problems established in [51, 47]. More precisely we show that the complexity of deciding the satisfiability of generelized (quantifier-free) formulas is characterized by the closure operations of the basis. Furthermore, in Chapter 8 we use a similar approach to prove that the learning complexity of generalized quantified formulas depends on the closure functions of the basis.

Furthermore, the set of closure operations of a basis constitutes a *clone*, which is a mathematical structure well studied in Universal Algebra. In consequence, it is possible to move results about clones into results about satisfiability or learnability of generalized formulas.

We focus on the study of the satisfiability of generalized (non-quantified) formulas and the learnability of quantified formulas.

1.2.1 Satisfiability of Generalized (quantifier-free) Formulas

The satisfiability of generalized (non-quantified) formulas, or Form(Γ) for a non-necessarily boolean basis Γ is related with a well-studied problem in complexity theory and artificial intelligence: the *constraint satisfaction problem*.

A constraint satisfaction problem is represented by a set of variables, a domain of values for each variable, and a set of constraints between variables. Generally, a constraint C is represented by two components: a *scope* S that expresses the set of variables constrained and a *relation* R which expresses the combination of values allowed for those variables. The aim of a constraint satisfaction problem is then to find an assignment of values to the variables that satisfies the constraints.

We are interested in a family of constraint satisfaction problems, denoted CSP(Γ), in which every constraint relation belongs to a fixed set Γ. This problem is precisely a reformulation of the problem of deciding the satisfiability of generalized formulas over Γ, Form(Γ).

There exists a large collection of individual results about the complexity of CSP(Γ) (see for example [15, 49, 52, 59, 65, 87, 88]). But to our knowledge, there has only been two attempts to study the complexity of CSP(Γ) in a uniform way. The first attempt, due to Feder and Vardi [31], uses an approach based on Datalog Programs and Group Theory and produced, among other important results, a classification of the known tractable classes in three families, called *width* 1, *bounded strict width*, and *subgroup problems*. The second attempt, was originally defined by Jeavons et al. [49, 50] and focuses on the algebraic properties of constraints. As we discussed above we follow the second approach in this thesis.

Let us resume briefly the so far known tractable families of basis in the framework of closure operations. As we have discussed above, the complexity of deciding the satisfiability of constraint satisfaction problems over a basis Γ is characterized by the set of closure operations of Γ, which is a clone. Thus, the main goal of the approach is to identify which sets of operations or clones guarantee tractability. Before this thesis, four tractable families of clones had been identified, each generated by a single operation.

(a) Near-unanimity operations, which characterizes exactly the class of bounded strict width problems.

(b) Affine operations, which is included in the class of subgroup problems.

(c) ACI operations, which is included in the class of width 1 problems.

(d) Constant operations, which is included in the class of width 1 problems.

We have extended the list of tractable clones of operations in several ways. It is convenient to remark that contrary to the previous tractable clones, some of the new tractable clones identified in this thesis are not generated by a single operation. This fact justifies the introduction of the notion of tractable set of operations, or tractable clone, instead of tractable operation which has been the customary case. Let us enumerate the new tractable cases:

1. Coset Generating Operations
Subgroup problems, as defined by Feder and Vardi [31] contains exactly the basis Γ such that every relation in Γ is the coset of some subgroup of the power group for a fixed group. In [51], an important subclass of subgroup problems is characterized in terms of closure functions. More precisely, in [51] it is shown that if the group is abelian, then the basis is preserved by an affine operation.

We extend this result by introducing a new family of operations, called coset generating operations which capture subgroup problems even in the non-abelian case. In other words, we introduce a family of closure operations that characterizes the class of subgroup problems identified in [31].

It is important to notice that Feder and Vardi [31] studied some ways to extend subgroup problems preserving tractability. These extensions are not covered by closure under coset generating operations.

2.- Width 1 *Problems*
The class of width 1 problems is identified in [31] using an approach based in Datalog Programs. Datalog programs turn out to be equivalent to a well known technique developed to solve constraint satisfaction problems: local consistency methods [61, 33, 34, 48, 28]. A local-consistency method is a procedure that adds to a constraint network all the constraints that appear implicitly in the problem up to a certain level. Since there exists problems which can not be solved by local consistency methods, there is a large body of work [35, 25, 28, 48] which derives conditions that guarantee that a problem can be solved by a certain local consistency method.

We define a notion of local consistency, called $(1, k)$-consistency, that can be regarded as the natural extension of the well known notion of arc-consistency [61] to non-binary problems. We say that a basis is width 1 if every problem in CSP(Γ) can be solved by enforcing $(1, k)$-consistency. Feder and Vardi [31] gave an alternative definition of constraint satisfaction problems in terms of Datalog Programs.

We give a characterization of width 1 problems in terms of closure functions. That is, we present an algebraic condition (in the sense of [51]) on the relations used to build constraint problems which ensure that any problem built using these relations can be solved in polynomial time by enforcing $(1, k)$-consistency for some k. Furthermore, we show that this condition is also necessary. Therefore we have an alternative characterization in terms of closure operations of the class of width 1 problems, identified in [31].

We revisit some classes already known to be tractable, such as ACI operations and constant operations [51]. We show that them are particular cases or width 1 problems obtaining an alternative proof of tractability. Finally, we introduce a new particular case of width 1 problem, called CSCC clone. A CSCC clone is generated by two operations, a binary commutative conservative operation and a constant semiprojection operation.

3.- Para-primal Algebras
Finally, in Chapter 7 we identify a new tractable family, called *para-primal algebras*. The class of tractable problems identified by para-primal algebras contains some problems that cannot be accounted for any of the classes identified by Feder and Vardi. Therefore, it constitutes a new family of tractable problems.

Para-primal algebras have been studied in Universal Algebra. We make an intensive use of algebraic results in the proof of the tractability of para-primal algebras. The intuition underlying the tractability of para-primal algebras is the following: Under some technical assumptions, a basis is preserved by a para-primal algebra if and only if the set of binary relations that can be "simulated" using relations in Γ is quite restricted. Thus, we can interpret the tractability of para-primal algebras in the following way: if the binary problems obtained using constraint relations from a basis are "simple enough", then the class of arbitrary problems using constraint relations in the basis is solvable.

Alternatively, a constraint satisfaction problem where the basis is closed under a para-primal algebra can be regarded as a non-trivial combination of some subgroup problems. Consequently, it is solved by dividing the constraint instance in its components and solving them independently.

As a consequence of the tractability of para-primal algebras and a very remarkable result in Universal Algebra concerning the classification of homogeneous algebras we prove a complete classification for constraint satisfaction problems under the restriction that the basis contains all the binary relations associated to permutations. This result generalizes a known previous result due to Cooper et al. [15] in several ways.

Summarizing our findings in the complexity of constraint satisfaction problems: (1) We give a characterization, in terms of closure operations, of the class of subgroup problems, defined in [31]. (2) We give also a characterization, in terms of closure opeations, of the class of width 1 problems defined in [31]. Using this characterization we identify a new tractable subclass of width 1 problems, called CSCC problems. (3) We identify a new tractable class imcomparable with those identified in in [31], called para-primal algebras. As a consequence of this result we get a dichotomy classification for the class of constraint satisfaction problems where permutations are available.

With results (1) and (2) we complete the characterization, in terms of closure operations, of the tractable classes identified in [31], finishing the research project started in [48]. Result (3) goes an step further by introducing a new class of tractable problems.

1.2.2 Learnability of Generalized Quantified Formulas with constants

Finally, in chapter 8 we address the problem of learning generalized quantified formulas with constants. Similarly to the study of constraint satisfaction problems we prove that the learning complexity of $\exists\forall\text{-}\,\mathrm{Form}_C(\Gamma)$ is characterized by an algebraic property of the basis: the clone of idempotent closure operations of Γ.

First, we study the problem of identifying learnable classes. We introduce a related representation class: subuniverses of power algebras for a fixed algebra. We introduce a very simple learning strategy based in a well known algorithm for intersection-closed classes: the closure algorithm [86]. Using this approach we identify three families, namely, coset generating operations, near-unanimity operations and para-primal algebras, which are learnable in the model of exact learning with equivalence queries [1].

These families of learnable cases cover all the learnable cases in the boolean domain. 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. Second, the total number of concepts with a determined number of attributes is a number singly exponential in the number of attributes. It is an interesting open question to clarify the relationship between this property and the learnability of quantified formulas.

Finally, we study whether the list of learnable classes is complete. We show that under different restrictions in the clone structure, namely, minimal, plain and homogeneous clones, we have a dichotomy. It is remarkable that the non-learnability results, similarly to the boolean case, are obtained in the powerful model of PAC-prediction with membership queries [3]. In consequence, we provide some (small) evidence that the gap between the known learnable classes and the complete list is not very wide.

1.2.3 Other Related Results

Finally we study some other applications of the connection between formulas and clones. In particular we make use of the fact that the structure of the clone lattice in the 2-element case is completely known [76] to derive some results about boolean formulas.

We start by giving alternative proofs of the results about satisfiability and learnability of generalized quantified formulas presented before, using the description of the clone lattice for the 2-element domain, also known as Post's lattice. The idea underlying this approach is that the task of proving a dichotomy or classification theorem over generalized quantified formulas can be splitted in two parts: one part involving complexity-theoretic concepts and another part concerning the expressivity power of classes of formulas. The key idea is that under some technical assumptions, clones capture the notion of expressivity power of a formula. In consequence, this part of the proof (which uses to be the more difficult) can be avoided referencing results in clone theory, specially in the boolean case, since the clone structure is completely known. Using this approach we provide much simpler proofs.

Furthermore, we present two new applications of Post's classification. We study the inverse satisfiability problem consisting in to decide whether there exists some formula F in a family of formulas such that the set of satistying assignments of F coincides with a given set of assignments. This problem has also been called *structure identification* [27] in the context of constraint satisfaction. In this setting, the problem can be interpreted as deciding whether a family of constraint networks is expressive enought to represent some piece of information. We provide several conditions that guarantee polynomial-time solvability of the problem. We make use of the fact that every clone in the boolean domain is finitely generated to derive a simple algorithm for solving in polynomial time the inverse satisfiability for generalized quantified boolean formulas over a fixed basis, extending the results in [54].

Finally, we study the computational complexity of learning (general) boolean formulas and circuits over some fixed basis Γ. This class of formulas coincides with the class of generalized quantified formulas in the fact that both have a basic set of relations (or functions, since in the boolean case relations and functions are equivalent) which are the elemental pieces that we use to build the formulas. They differ in the way this basic functions (or relations) are used: whereas in a generalized quantified formula we only allow conjunction of this basic functions and quantification, in a (general) formula we use the basic functions to compose formulas.

We show that the learning complexity depends only on the clone generated by the basis of functions used in the gates of the formula (circuit). More precisely we show that, under some crytographic assumptions, for every such basis Γ, the set of formulas with gates in Γ is either polynomial-time learnable in the exact learning model with equivalence queries alone or membership queries alone or not learnable in the prediction with membership model. Furthermore, we show which bases fall in each type. More precisely we show that the class of formulas over basis Γ is polynomial-time learnable if and only if:

- Every function in Γ is identically true or it can be expressed as a disjunction of some variables.

- Every function in Γ is identically false of it can be expressed as a conjunction of some variables.

- Every function in Γ can be expressed as a linear equation over the two-element field $\{0, 1\}$.

The structure of the thesis is as follows:

- **Chapter 2.** We define formally the class of generalized formulas and we introduce some preliminaries that will be used intensively along the rest of the thesis.
- **Chapter 3.** We study the learnability problem for generalized quantified boolean formulas with constants. Results in this chapter have appeared in [18, 20].
- **Chapter 4.** We adress the problem of extending results on generalized quantified boolean formulas with constants to generalized constant-free quantified boolean formulas. We solve it for satisfiability and learnability problems. Results in this chapter have appeared in [19].
- **Chapter 5.** We introduce the basic algebraic concepts for the following chapters. No complexity notions are involved. We present a proof of the galois connection between sets of relations and operations that differs in some aspects from the previous ones. The rest of the material is not original with the exception of the section devoted to the family of coset generating operations and its properties.
- **Chapter 6.** We present the relationship between constraint satisfaction problems and closure operations. We show a characterization of width 1 problems. Results in this chapter have appeared in [24].
- **Chapter 7.** We study the tractable class of para-primal algebras. Results in this chapter appear in [22]
- **Chapter 8.** We study the learnability of generalized quantified formulas. Results in this chapter have appeared in [23].
- **Chapter 9.** We present some applications of Post's classification over computational problems on boolean formulas. More precisely, we give

simpler proof of results in Chapters 3 and 4. Furthermore, we study the inverse satisfiability problem and the learning complexity of (general) boolean formulas and circuits. Results in this chapter have appeared in [21].

Chapter 2

Preliminaries

In this chapter we introduce some concepts that will be intensively used in next chapters. In particular, we define the class of generalized formulas and its variants which plays a major role in this thesis.

2.1 Logical Preliminaries

Let $V = \{x_1, x_2, \dots\}$ be a countably 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 sequences or tuples of elements in D. An *assignment* or *string* over D is a tuple in D^*. If t is an assignment over D , $|t|$ denotes its length, and for every element $d \in D$, $|t|_d$ denotes the number of occurrences of d in t. There exists an unique assignment of length 0, we call it λ. 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. Given two assignments t_1 and t_2, $t_1 t_2$ denotes the assignment obtained concatenating t_1 and t_2. 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.

The domain containing exactly two elements, i.e., $\{0, 1\}$, is called boolean domain and denoted $\mathcal{B}$. Value 0 corresponds to *false* and value 1 is corresponds to *true*. Relations over the boolean domain are called *boolean* or *logical* relations. Strings (assignments) over boolean domains are called *binary* strings (assignements). Logical operators ($\vee$, $\wedge$, $\neg$, $\oplus$, $\rightarrow$) can also be applied to binary assignments meaning that they are operated componentwise. Given two boolean assignements t_1, t_2 of the same length n, operator $\leq$ is defined as

$$t_1 \leq t_2 \iff \forall 1 \leq i \leq n \ \ (t_1[i] \rightarrow t_2[i]).$$

2.1.1 Formulas

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

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 is a relation of 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 $\exists\forall\text{-}\,\mathrm{Form}_C(\Gamma, D)$, is the smallest set of formulas such that:

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

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

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

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

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 $\exists\text{-}\,\mathrm{Form}_C(\Gamma)$. 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 $\mathrm{Form}_C(\Gamma)$. Notice that the set of constants allowed depends on the domain of the basis Γ.

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 $\exists\forall\text{-}\,\mathrm{Form}(\Gamma)$, $\exists\text{-}\,\mathrm{Form}(\Gamma)$, $\mathrm{Form}(\Gamma)$ 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 $\mathrm{Rel}(\Gamma) = \{[\Phi] : \Phi \in \mathrm{Form}(\Gamma)\}$. Similarly, we also define $\exists\forall\text{-}\mathrm{Rel}(\Gamma)$, $\exists\text{-}\mathrm{Rel}(\Gamma)$, $\mathrm{Rel}_C(\Gamma)$, $\exists\forall\text{-}\mathrm{Rel}_C(\Gamma)$, and $\exists\text{-}\mathrm{Rel}_C(\Gamma)$.

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

2.2 Some Classes of Relations

We use the term *clause* in a broad sense to mean any boolean relation in a conjunction. A Conjunctive Normal Formula (CNF) is a conjunction of disjunctions of literals. A CNF in which every clause has at most k-literals is referred to as a k-CNF.

The following types of clauses will be used in the next chapters:

- A clause is *bijunctive* if it is the disjunction of at most 2 literals.
- A clause is *horn* (resp. *antihorn*) if it is the disjunction of at most one affirmed (resp. negated) variables. A horn clause can be regarded as a rule of the type $(y_1 \ldots y_k \to y_{k+1})$ where every y_i is a variable or a boolean constant.
- A clause is *k-weakly monotone* (resp. *k-weakly antimonotone*) $(k \geq 2)$ if it is the disjunction of at most k affirmed (resp. negated) variables or the disjunction of at most two literals with at most one negated (resp. affirmed) variable. That is, a k-weakly antimonotone clause is a horn clause where we only allow rules like $(y_1 \ldots y_k \to 0)$ and $(y_l \to y_j)$ where every y_i is a variable or a constant. A clause is *weakly monotone* (resp. *weakly antimonotone* if it is k-weakly monotone (resp. k-weakly antimonotone) for some $k \geq 2$.

- A clause is *monotone* (resp. *antimonotone*) if it is the disjunction of affirmed variables (resp. negated variables).

- A clause is *equality monotone* (resp. *equality antimonotone*) if it is the equality between two variables or it is the disjunction of affirmed variables (resp. negated variables).

- A clause is *affine* if it is an equation over the two-element field $\{0, 1\}$.

- A clause of rank k is 0-*valid* (resp. 1-*valid*) if it is empty or it is satisfied by 0^k (resp. 1^k).

- A clause is 0-*affine* if it is affine and 0-valid.

- A clause is 1-*affine* if it is affine and 1-valid.

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

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

2.3 Computational Problems over Generalized Formulas

We will focus in two computational problems over generalized formulas: satisfiability and learnability problems.

2.3.1 Satisfiability Problems

The problem of determining whether a given formula is satisfiable has been widely studied. A *solution* of a formula Φ is any assignment over the free variables in Φ that satisfies Φ.

Definition 1 *For any set of formulas $\mathcal{F}$,* SAT($\mathcal{F}$) *is defined to be the class of decision problems with:*

- *Instance: A formula Φ in $\mathcal{F}$.*
- *Question: Does Φ have a solution?*

A class of formulas $\mathcal{F}$ is said to be *tractable* if there exists an algorithm that solves SAT($\mathcal{F}$) with running-time polynomial in the size of the representation.

The satisfiability problem for propositional formulas is the most studied. It is well known that even restricted to formulas in conjunctive normal form with at most three literals per clause the problem is NP-complete [13]. Therefore, researchers have attempted to determine the satisfiability of subclasses of propositional boolean formulas, in particular inside CNF. For example, the satisfiability in polynomial time of bijunctive formulas was shown in [13] and can be done also in linear time [70]. The satisfiability of Horn formulas and read-twice CNF formulas have also been shown to be decidable in linear time [29, 71].

The problem of deciding the satisfiability of quantified boolean formulas is PSPACE-complete [82, 83] in general and some effort has been devoted to classify the polynomial-time decidable classes. A linear time algorithm when all clauses have at most two variables can be found in [4]. In [53] a cubic time algorithm for quantified Horn formulas has been showed.

Schaefer [81] studied the satisfiability of some generalized boolean formulas as a unifying framework to study the complexity of satisfiability problems when some restriction in the relations allowed to build formulas are placed. More precisely, Schaefer [81] studied and solved completely the following problems: $\mathrm{SAT}(\mathrm{Form}(\Gamma))$,$\mathrm{SAT}(\mathrm{Form}_C(\Gamma))$, and $\mathrm{SAT}(\exists\forall\text{-}\,\mathrm{Form}_C(\Gamma))$, for finite boolean basis Γ.

In this thesis we further continue that line of reseach by extending Schaefer's results to different classes of generalized formulas. We will study the satisfiability of constant-free quantified boolean formulas ($\mathrm{SAT}(\exists\forall\text{-}\,\mathrm{Form}(\Gamma))$, Γ a finite set of boolean relations) completing the knowledge about generalized boolean formulas. Furthermore, we study the satisfiability problem for generized formulas over larger domains ($\mathrm{SAT}(\mathrm{Form}(\Gamma))$), known as constraint satisfaction problem.

Let us state formaly Schaefer's classification results. The most known Schaefer's result is the one concerning the satisfiability of generalized formulas over a finite boolean basis.

Theorem 1 *[81](Schaefer's Dichotomy Theorem) Let* Γ *be a finite set of boolean relations. If* Γ *satisfies one of the conditions* (a)-(f) *below, then* $\mathrm{SAT}(\mathrm{Form}(\Gamma))$ *is polynomial-time decidable. Otherwise,* $\mathrm{SAT}(\mathrm{Form}(\Gamma))$ *is log-complete in NP.*

a.- Γ *is bijunctive.*

b.- Γ *is affine.*

c.- Γ *is horn.*

d.- Γ *is antihorn.*

e.- Γ *is* 0*-valid.*

f.- Γ *is* 1*-valid.*

Schaefer [81] also showed a dichotomy theorem for the satisfiability of the class $\mathrm{Form}_C(\Gamma)$. The cases to consider when constants are available are simpler than in its constant-free counterpart. In fact, the dichotomic result for the satisfiability of $\mathrm{Form}_C(\Gamma)$ was used, as an intermediate step, in the proof of the more general result presented above. The dichotomy for generalized boolean formulas with constants is as follows:

Theorem 2 *[81] Let* Γ *be a finite set of boolean relations. If* Γ *satisfies one of the conditions* (a)-(d) *below, then* $\mathrm{SAT}(\mathrm{Form}_C(\Gamma))$ *is polynomial-time decidable. Otherwise,* $\mathrm{SAT}(\mathrm{Form}_C(\Gamma))$ *is log-complete in NP.*

a.- Γ *is bijunctive.*

b.- Γ *is affine.*

c.- Γ *is horn.*

d.- Γ *is antihorn.*

Schaefer also proved a dichotomy theorem for the satisfiability of generalized quantifed boolean formulas with constants.

Theorem 3 *[81] Let Γ be a finite set of boolean relations. If Γ satisfies one of the conditions (a)-(d) below, then* $\mathrm{SAT}(\exists\forall\text{-}\mathrm{Form}_C(\Gamma))$ *is polynomial-time decidable. Otherwise,* $\mathrm{SAT}(\exists\forall\text{-}\mathrm{Form}_C(\Gamma))$ *is log-complete in PSPACE.*

a.- Γ *is bijunctive.*

b.- Γ *is affine.*

c.- Γ *is horn.*

d.- Γ *is antihorn.*

It is remarkable that in all the cases, there is a sharp dichotomy between solvable and non-solvable problems. Furthermore, notice that the tractable cases for generalized quantified boolean formulas and generalized (quantifier-free) boolean formulas are the same whenever constants are available.

2.3.2 Learning Problems

Another extension of Schaefer's result that we will study is the learnability of generalized formulas. The following sections introduce the basic concepts and models of computational learning that we will use.

Representation class

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.

Along these pages we use three models of learning, all of them fairly standard: Angluin's model of exact learning with queries defined by Angluin [1], Valiant's model of PAC learning [86], and the model of PAC-prediction with membership queries as defined by Angluin and Kharitonov [3].

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 learns exactly $\mathcal{C}$.

The PAC Learning Model

Let $\mathcal{H}$ be a representation class. A *PAC algorithm with membership queries*, 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 two 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 may make any number of membership queries or requests for random classified examples. 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. *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 *learns* 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 outputs a concept name v such that the probabity that the classification of a string x, chosen randomly according to $\mathcal{D}$, by c, concidides with the classification of x by $K_{\mathcal{C}}(v)$ is at least $1 - \epsilon$.

A representation of concepts $\mathcal{C}$ is *PAC learnable with membership queries* if and only if there is a PAC-algorithm A that runs in polynomial time and learns $\mathcal{C}$. If A does not ask membership queries we say that $\mathcal{C}$ is *PAC learnable.*

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 the unknown 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 $\mathcal{C}$.

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 [74] *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 thesis 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.*

Relationship between Learning Models

So far we have presented three main learning models: exact learning, PAC leaning and PAC-prediction. Every model can be endowed with membership queries. Thus we have six different learning models, namely, exact learning with equivalence queries (EQ), exact learning with equivalence and membership queries (EQ+MEM), PAC learning (PAC), PAC learning with membership queries (PAC+MEM), PAC prediction (PRED), and PAC prediction with membership queries (PRED+MEM), satisfying certain relationships.

Informally, given two learning models $\mathcal{M}$, $\mathcal{N}$ we say that $\mathcal{N}$ is stronger than $\mathcal{M}$ if every representation class learnable in model $\mathcal{M}$ is also learnable in model $\mathcal{N}$. For instance, every learning model with membership queries is stronger than its membership-free counterpart.

Furthermore, given any algorithm A using equivalence queries, it is possible to replace every equivalence query by a certain amount of requests of labelled examples, as shown in [1]. Thus, PAC learning is stronger than exact learning with equivalence queries.

It is also possible to convert any PAC learning algorithm into a PAC prediction algorithm by keeping the output concept as an internal representation an using it to answer the classification of the element to predict [40]. Thus, PAC predition is stronger than PAC learning.

Diagram 2.3.2 shows the relationship beetwen the learning models. Arrows in the diagram should be interpreted in the following way: $A \longrightarrow B$ means that model B is stronger that model A. For clarity, only direct relationships are depicted. To obtain a full picture of the relationships apply transitive closure to the diagram.

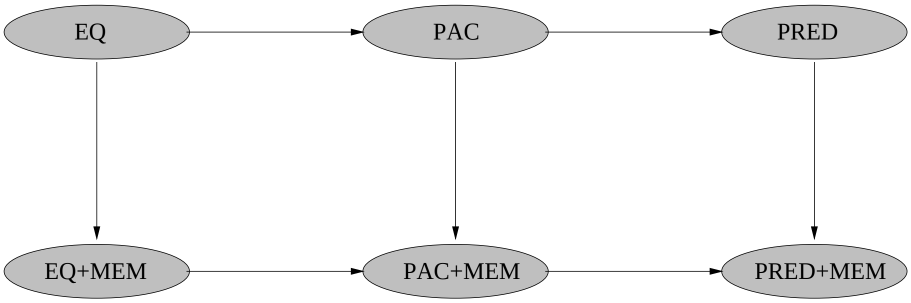

Figure 2.1: Relationship between learning models

Chapter 3

Learnability of Quantified Boolean Formulas

3.1 Introduction

In this chapter we study the learning complexity of quantified formulas with constants over a finite boolean basis ($\exists\forall\text{-}\,\mathrm{Form}_C(\Gamma)$), as defined in Section 2.1.1.

In despite of the considerable amount of results about the learnability of propositional formulas [1, 2, 8, 46], first-order formulas have not received much attention (see [60] and further references in that paper). An active line of research in predicate logics is for instance *Inductive Logic Programming* (ILP) [67, 68, 58].

We restrict ourselves to the boolean basis mainly because the boolean case does not exhibit the combinatorial explosion typical of larger domains and, in consequence, it is possible to study the problem using exclusively "elementary methods". Another important reason to consider boolean formulas is that allows to establish comparison with some other already known results about quantified boolean formulas. For example, from Schaefer's Dichotomy Theorem for quantified formulas (Theorem 3) and the Dichotomy Theorem of this chapter (Theorem 4) it can be inferred that, in this framework, learnability is slightly harder than satisfiability.

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

Furthermore, we give an interesting classification of the polynomially learnable classes. We show that, in a sense that will be precised later, $\exists\forall$- $\mathrm{Form}_C(\Gamma)$ is polynomially learnable if and only if Γ is bijunctive, weakly monotone, weakly antimonotone or affine.

It is interesting to compare this classification with some previous known results about the learnability of quantifier-free formulas. First, notice that whereas k-CNF are polynomially learnable with equivalence queries, the equivalent result for quantified formulas is only valid for $k \leq 2$. For larger arity ($k \geq 3$), quantified k-CNF's are learnable only in the particular cases of k-weakly monotone and k-weakly antimonotone formulas.

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

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

3.2 The Dichotomy Theorem

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

Theorem 4 *(Dichotomy Theorem for the learnability of quantified boolean formulas with constants) Let* Γ *be a finite set of boolean relations. If* Γ *satisfies one of the conditions (a)-(d) below, then the class* $\mathcal{C}_{\exists\forall\text{-}\mathrm{Form}_C(\Gamma)}$ *is polynomially exactly learnable with improper equivalence queries. Otherwise,* $\mathcal{C}_{\exists\text{-}\mathrm{Form}_C(\Gamma)}$ *is not polynomially predictable with membership queries under the assumption that public key encryption systems secure against chosen ciphertext attack exist.*

a.- Γ *is bijunctive.*

b.- Γ *is affine.*

c.- Γ *is weakly monotone.*

d.- Γ *is weakly antimonotone.*

We refer the reader to Angluin and Kharitonov [3] for definitions of the cryptographic concepts.

Schaefer [81] (Theorem 3) proves a similar dichotomy theorem for the satisfiability of $\exists\forall\text{-}\mathrm{Form}_C(\Gamma)$. He shows that this problem is polynomial-time solvable if and only if Γ is bijunctive, horn, antihorn or affine. Otherwise the problem is PSPACE-complete (NP-complete if we take only $\exists\text{-}\mathrm{Form}_C(\Gamma)$). We can note here that if a basis Γ does not fall in the classes in Schaefer's theorem then the representation of concepts $\exists\forall\text{-}\mathrm{Form}_C(\Gamma)$ is not polynomial-time evaluable, i.e., given an example and a representation of a concept, it is not possible to decide in polynomial time whether the example is a member of the concept, unless P=PSPACE.

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

Observe that for the negative results only existential quantifiers are necessary. Therefore the main theorem does not change at all if we are restricted to formulas without universal quantifiers.

Along the rest of this chapter, a set of boolean relations Γ will be called a *basic set* if Γ is bijunctive, weakly monotone, weakly antimonotone, or affine and the representation of concepts $\exists\forall$- $\mathrm{Form}_C(\Gamma)$ formed from a basic set Γ will be called *basic representation of concepts*.

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

a.- The efficient learnability of basic classes follows from the following results:

 (i) the expressive power of formulas over basic sets is essentially the same with and without using quantifiers (up to polynomials in size), and

 (ii) in consequence the learning problem can be reduced to quantifier-free formulas that are contained in some classes already known to be learnable.

b.- For the non-learnability results, we show that

 (i) the quantified formulas over any non-basic set can express a double implication $[(x \wedge y) \rightarrow z]$ or its dual.

 (ii) Then we show that this implication is enough to simulate boolean circuits, which are hard to learn under cryptographic assumptions.

The rest of the chapter is structured as follows:

Section 3.2.1 contains exclusively results about relations and their expressivity power. No learning notions are involved. This section contains items **(ai)** and **(bi)**. It starts with some characterizations of some classes in semantic terms, that is, in terms of what elements are in the relation, rather that in terms of defining formulas as in the definitions. These alternative semantic definitions are easier to handle to prove item **(bi)**. Finally, item **(aii)** is shown in Section 3.2.2 and Section 3.2.3 contains item **(bii)**.

3.2.1 Results in Logic

Closure under Quantification

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

Let us state the main result of this section:

Lemma 3 *If Γ is a bijunctive (resp. k-weakly monotone, k-weakly antimonotone, affine) set of boolean relations then $\exists\forall\text{-}\,\mathrm{Form}_C(\Gamma)$ is bijunctive (resp. k-weakly monotone, k-weakly antimonotone, affine).*

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

We begin with some claims of easy proof:

Claim 1 *Let $x \in V$ be a variable. If Φ is a bijunctive (resp. k-weakly monotone, k-weakly antimonotone, affine) formula then $\Phi\left[\begin{smallmatrix}x\\0\end{smallmatrix}\right]$ and $\Phi\left[\begin{smallmatrix}x\\1\end{smallmatrix}\right]$ are bijunctive (resp. k-weakly monotone, k-weakly antimonotone, affine).*

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

Claim 2 *Let $x \in V$ be a variable. Let C_x and $C_{\overline{x}}$ be a pair of clauses that contain the literal x and $\overline{x}$ respectively. If C_x and $C_{\overline{x}}$ are bijunctive (resp. k-weakly monotone, k-weakly antimonotone), then $\mathcal{R}(C_x, C_{\overline{x}}, x)$ is bijunctive (resp. k-weakly monotone, k-weakly antimonotone).*

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

Claim 3 *For every variable $x \in V$ and for every formula Φ, $\forall x \Phi \equiv \Phi\left[\begin{smallmatrix}x\\0\end{smallmatrix}\right] \wedge \Phi\left[\begin{smallmatrix}x\\1\end{smallmatrix}\right]$.*

Claim 4 *For every variable $x \in V$ and for every CNF formula Φ such that $\Phi \equiv \bigwedge_{C \in G_x \cup G_{\overline{x}} \cup G} C$ where G_x, $G_{\overline{x}}$, and G are sets of clauses that contain the literal x, the literal $\overline{x}$, and none of them respectively, the following equivalence holds: $\exists x \Phi \equiv \bigwedge_{C_x \in G_x, C_{\overline{x}} \in G_{\overline{x}}} \mathcal{R}(C_x, C_{\overline{x}}, x) \wedge \bigwedge_{C \in G} C$.*

From Claims 1, 2, 3, and 4 we can derive the following lemma:

Lemma 4 *Let Φ be a bijunctive (resp. k-weakly monotone, k-weakly antimonotone) formula. Then, for every variable $x \in V$, $\forall x \Phi$ and $\exists x \Phi$ are both bijunctive (resp. k-weakly monotone, k-weakly antimonotone) formulas.*

The analogous result for affine functions is proved differently.

Lemma 5 *Let Φ be an affine formula. Then for every variable $x \in V$, $\forall x \Phi$ and $\exists x \Phi$ are both affine.*

Proof. Let Φ be equivalent to a system of linear equations over the two element field $\{0, 1\}$. To prove the case $\forall x \Phi$ we only need to apply Claims 1 and 3 or, even easier, note that $\forall x \Phi \equiv 0$ if x appears in Φ, and $\forall x \Phi \equiv \Phi$ otherwise. So we only need to consider the case $\exists x \Phi$. We can take some equation that contains the variable x, separate x in one side of the equation and substitute the other side in the rest of equations. This process gives us a system of equations equivalent to $\exists x \Phi$. ■

We can eliminate systematically the quantifiers from inside to outside preserving the bijunctivity (resp. k-weakly monotonicity, k-weakly antimonotonicity, affinity) obtaining Lemma 3

Semantic Characterizations

In this section we give semantic characterizations of some sets of boolean relations. Other semantic characterizations can be found in Schaefer [81]. Semantic characterizations are more convenient to our purposes than the definitions given in Section 2.2. This is because, as we will see in the next section, we are interested in the boolean relations generated by some relation R that does *not* belong to a particular class. In these cases, using the semantic characterizations we can infer that there exists some assignment in R not satisfying determined property and establish consequences from this fact.

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

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

$$\mathrm{pr}_T\, R = \{\mathrm{pr}_T\, t : t \in R\}$$

Notice that a relation equivalent $\mathrm{pr}_T\, R$ can be obtained with the formula: $[R = [\exists x_{l_1}, \ldots, \exists x_{l_m} R(x_1, \ldots, x_k)]$ where $\{l_1, \ldots, l_m\}$ is the set of indices not in T. Therefore, $\mathrm{pr}_T\, R \in \exists\text{-}\,\mathrm{Rel}_C(\{R\})$.

We have $\mathrm{pr}_\emptyset\, t = \lambda$ and $\mathrm{pr}_\emptyset = R$ is the relation 1 if $R \neq \emptyset$ and 0 otherwise.

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

The notion of j-compatibility is the key to characterize weakly antimonotone boolean relations:

Lemma 6 *Let R be a boolean relation of rank k. The following conditions are equivalent:*

a.- R is weakly antimonotone.

b.- For every $T \subseteq \{1, \ldots, k\}$ and every assignment $t \in \{0,1\}^{|T|}$ not in $\mathrm{pr}_T\, R$ and $|T|$-compatible with $\mathrm{pr}_T\, R$:

  i.- $|t|_0 = 0$, or

  ii.- $|t|_0 = 1$ and $|t|_1 \le 1$.

c.- For every $T \subseteq \{1, \ldots, k\}$ and every assignment $t \in \{0,1\}^{|T|}$ not in $\mathrm{pr}_T\, R$ and $|T|$-compatible with $\mathrm{pr}_T\, R$:

  i.- $|t|_0 \le 1$, and

  ii.- If $|t|_0 = 1$ then $|t|_1 \le 1$.

[1] The notion of compatibility used in this chapter differs only slightly from the compatibility defined by Kavvadias and Sideri [54]. Precisely, the j-compatibility corresponds exactly to the $(j-1)$-compatibility in the sense of [54].

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

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

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

where

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

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

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

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

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

    Step 2 If t_i is $|T_i|$-compatible with $\mathrm{pr}_{T_i} R$ then **stop**, otherwise there exists some proper subset $T \subset T_i$ such that $\mathrm{pr}_T t_i \notin \mathrm{pr}_T R$. Assign $t_{i+1} := \mathrm{pr}_T t_i$, $T_{i+1} := T$, $i := i + 1$; go to step 2.

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

- **[a⇒b]** If R is a weakly antimonotone boolean relation then for every $T \subseteq \{1, \dots, k\}$, by Lemma 3, $\mathrm{pr}_T R$ is a weakly antimonotone boolean relation. Let $\Phi = \bigwedge_{j=1}^{s} C_j$ be a CNF formula where each clause is weakly antimonotone such that $[\Phi] = \mathrm{pr}_T R$. Let $t \notin \mathrm{pr}_T R$ be a $|T|$-compatible with $\mathrm{pr}_T R$ assignment. By the $|T|$-compatibility, t cannot falsify any clause of less than $|T|$ literals, and therefore must falsify a clause of exactly $|T|$ literals C_l. If C_l is of type (i) or (ii) in the definition of weakly antimonotone clauses then t satisfies the conditions (bi) or (bii) respectively. ■

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

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

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

a.- R is horn.

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

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

Proof. Equivalence between conditions (a) and (b) is well known [45]. Equivalence between condition (a) and (b) is obtained adapting the proof of Lemma 6. ■

Main Result in Logic

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

Theorem 5 *Let Γ be a finite non-basic set of boolean relations, then $\{[x \vee y \vee \overline{z}], [\overline{x} \vee \overline{y} \vee z]\} \cap \exists\text{-}\mathrm{Rel}_C(\Gamma) \neq \emptyset$.*

The remainder of this section is devoted to the proof of Theorem 5. The following results are from [81].

Lemma 8 *[81] Let R be a boolean relation which is not horn, then* $\{[x \not\equiv y], [x \vee y]\} \cap \exists\text{-}\operatorname{Rel}_C(\{R\}) \neq \emptyset$.

Lemma 9 *[81] Let R be a boolean relation which is not antihorn, then* $\{[x \not\equiv y], [\overline{x} \vee \overline{y}]\} \cap \exists\text{-}\operatorname{Rel}_C(\{R\}) \neq \emptyset$.

Corollary 1 *[81] Let Γ be a finite set of relations which is not horn and not antihorn, then* $[x \not\equiv y] \in \exists\text{-}\operatorname{Rel}_C(\Gamma)$.

Lemma 10 *[81] Let Γ be a finite set of boolean relations which is not affine and not bijunctive, then $\exists\text{-}\operatorname{Rel}_C(\Gamma \cup \{[x \not\equiv y]\})$ is the set of all boolean relations.*

The following lemmas will apply when conditions (ci) or (cii) in Lemma 6 fail.

Lemma 11 *Let R be a boolean function of rank $k \geq 2$. Suppose that there is a k-compatible with R assignment $t \notin R$ that contains at least two zeroes. Then* $\{[x \not\equiv y], [x \vee y]\} \cap \operatorname{Rel}_C(\{R\}) \neq \emptyset$.

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

Lemma 12 *Let R be a boolean relation of rank $k \geq 3$. Suppose that there is a k-compatible with R assignment $t \notin R$ that contains exactly one 0. Then* $\{[\overline{x} \vee \overline{y} \vee z], [x \not\equiv y], [x \vee y]\} \cap \exists\text{-}\operatorname{Rel}_C(\{R\}) \neq \emptyset$.

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

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

For all $t' \in \{0,1\}^k$ such that $t' \neq t$ and $t'[i] = 0$, t' can be expressed as $t' = \bigwedge_{j \neq i, t'[j]=0} t_{j \leftarrow 0}$, so we can assume $t' \in R$, otherwise by Lemma 7 R is not horn and by Lemma 8 $\{[x \not\equiv y], [x \vee y]\} \cap \exists\text{-}\operatorname{Rel}_C(\{R\}) \neq \emptyset$.

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

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

The following result follows from Lemmas 11 and 12.

Lemma 13 *Let R be a boolean relation of rank k that is not weakly antimonotone. Then* $\{[x \not\equiv y], [\overline{x} \vee \overline{y} \vee z], [x \vee y]\} \cap \exists\text{-}\mathrm{Rel}_C(\{R\}) \neq \emptyset$.

Proof. Let $T \subseteq \{1, \ldots, k\}$ and $t \notin \mathrm{pr}_T R$ be an $|T|$-compatible with $\mathrm{pr}_T R$ assignment such that condition (c) in Lemma 6 is falsified. There are two cases according to what condition is falsified. If condition (ci) is falsified then t contains at least two zeros and by Lemma 11 $\{[x \not\equiv y], [x \vee y]\} \cap \mathrm{Rel}_C(\{\mathrm{pr}_T R\}) \neq \emptyset$. If condition (cii) is falsified then by Lemma 12 $\{[x \not\equiv y], [\overline{x} \vee \overline{y} \vee z], [x \vee y]\} \cap \exists\text{-}\mathrm{Rel}_C(\{\mathrm{pr}_T R\}) \neq \emptyset$. ■

By duality, we have:

Lemma 14 *Let R be a boolean relation of rank k that is not weakly monotone. Then* $\{[x \not\equiv y], [x \vee y \vee \overline{z}], [\overline{x} \vee \overline{y}]\} \cap \exists\text{-}\mathrm{Rel}_C(\{R\}) \neq \emptyset$.

Directly from Lemmas 13 and 14 we have:

Corollary 2 *Let Γ be a finite set of boolean relations that is not weakly monotone and not weakly antimonotone, then* $\{[x \not\equiv y], [\overline{x} \vee \overline{y} \vee z], [x \vee y \vee \overline{z}]\} \cap \exists\text{-}\mathrm{Rel}_C(\Gamma) \neq \emptyset$.

And now Theorem 5 follows from Corollary 2 and Lemma 10. ■

3.2.2 Basic Classes are Polynomially Exactly Learnable

In this section we show efficient learnability results for the basic classes. We use the closure under quantification shown in section 3.2.1 to prove that in a certain sense each of these classes is embedded in a known learnable class and therefore learnable using the algorithm for the more general class. We start by a formal definition of embedding between classes and a simple observation:

Definition 3 Let $\mathcal{C}, \mathcal{C}'$ be representations of concepts. We say that $\mathcal{C}$ is *embedded* in $\mathcal{C}'$ if there exists a polynomial p such that for every concept name $u \in X$ there exists some concept name $u' \in X$ such that $|u'| \leq p(|u|)$ and $K_{\mathcal{C}}(u) = K_{\mathcal{C}'}(u')$.

Observation 1 *Let $\mathcal{C}$ be a representation of concepts which is polynomially exactly learnable with equivalence queries in $\mathcal{H}$. For every representation of concepts $\mathcal{C}'$ embedded in $\mathcal{C}$, $\mathcal{C}'$ is polynomially exactly learnable with equivalence queries in $\mathcal{H}$.*

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

Theorem 6 *[1] For all integers $k \geq 0$ the class $\mathcal{C}_{k-\mathrm{CNF}}$ is polynomially exactly learnable with $O(n^k)$ proper equivalence queries.*

Theorem 7 *[9] Let be $\mathcal{C}_{Af}$ be the set of formulas formed by conjunctions of equations over the two-element field $\{0,1\}$. The class $\mathcal{C}_{Af}$ is polynomially exactly learnable with $n+1$ proper equivalence queries.*

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

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

- For every concept name u in $\mathcal{C}_{Af}$, $|u| \in O(n^2 \log n)$.

If all the concepts in a representation class $\mathcal{C}$ have size polynomial in n then we do not need to worry about the length of the representation in order to show that any representation class is embedded in $\mathcal{C}$, since this condition is satisfied automatically. From this observation and from Lemma 3 we can derive the following results:

Claim 5 *Let Γ be a finite set of boolean relations. The following conditions hold:*

- *If Γ is a bijunctive set of boolean relations then $\mathcal{C}_{\exists\forall\text{-}\mathrm{Form}_C(\Gamma)}$ is embedded in $\mathcal{C}_{2-\mathrm{CNF}}$.*
- *If Γ is a k-weakly monotone set of boolean relations then $\mathcal{C}_{\exists\forall\text{-}\mathrm{Form}_C(\Gamma)}$ is embedded in $\mathcal{C}_{k-\mathrm{CNF}}$.*
- *If Γ is a k-weakly antimonotone set of boolean relations then $\mathcal{C}_{\exists\forall\text{-}\mathrm{Form}_C(\Gamma)}$ is embedded in $\mathcal{C}_{k-\mathrm{CNF}}$.*
- *If Γ is an affine set of boolean relations then $\mathcal{C}_{\exists\forall\text{-}\mathrm{Form}_C(\Gamma)}$ is embedded in $\mathcal{C}_{Af}$.*

We are now ready for the positive learnability results. From the previous claim and Observation 1 we can derive the following theorem:

Theorem 8 *Let Γ be a finite set of boolean relations. The following conditions hold:*

- *If Γ is a bijunctive set of boolean relations then $\mathcal{C}_{\exists\forall\text{-}\mathrm{Form}_C(\Gamma)}$ is polynomially exactly learnable with $O(n^2)$ equivalence queries in $\mathcal{C}_{2-\mathrm{CNF}}$.*
- *If Γ is a k-weakly monotone set of boolean relations then the class $\mathcal{C}_{\exists\forall\text{-}\mathrm{Form}_C(\Gamma)}$ is polynomially exactly learnable with $O(n^k)$ equivalence queries in $\mathcal{C}_{k-\mathrm{CNF}}$.*
- *If Γ is a k-weakly antimonotone set of boolean relations then the class $\mathcal{C}_{\exists\forall\text{-}\mathrm{Form}_C(\Gamma)}$ is polynomially exactly learnable with $O(n^k)$ equivalence queries in $\mathcal{C}_{k-\mathrm{CNF}}$.*
- *If Γ is an affine set of boolean relations then $\mathcal{C}_{\exists\forall\text{-}\mathrm{Form}_C(\Gamma)}$ is polynomially exactly learnable with $n+1$ equivalence queries in $\mathcal{C}_{Af}$.*

3.2.3 Boolean Circuits are pwm-reducible to Formulas over Non-basic Classes

In this section we show that, in a sense made precise later, quantified boolean formulas from non-basic sets of relations are not learnable. We showed in Section 3.2.1 that non-basic sets can express the double implication $[(x \wedge y) \rightarrow z]$

or its dual counterpart. In this section we complete the proof by showing that sets of quantified formulas containing the double implication can simulate boolean circuits which are known to be not learnable. Simulation in the model of PAC-prediction with membership queries is characterized by the notion of *pwm-reduction*, as defined in Section 2.3.2 .

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

We consider the input variables as gates of fan-in 0. In [3] it is shown that under some cryptographic assumptions boolean circuits are not polynomially predictable with membership queries:

Theorem 9 *[3] If there exist public key encryption systems secure against chosen ciphertext attack, then $\mathcal{C}_{BC}$ is not polynomially predictable with membership queries.*

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

Lemma 15 $\mathcal{C}_{BC} \leq_{pwm} \mathcal{C}_{MBC}$.

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

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

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

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

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

where v_i denotes the value of gate i.

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

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

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

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

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

Lemma 16 $\mathcal{C}_{MBC} \leq_{pwm} \exists\text{-}\mathrm{Form}_C(\{\overline{x} \vee \overline{y} \vee z\})$

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

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

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

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

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

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

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

This reduction is very similar to the reduction from the evaluation of monotone boolean circuit problem to the horn satisfiability problem, used to show P-completeness of the latter problem (see, for example [38]).

By duality we have:

Lemma 17 $\mathcal{C}_{MBC} \leq pwm \; \mathcal{C}_{\exists\text{-}\mathrm{Form}_C(\{x \vee y \vee \overline{z}\})}$

We put the previous lemmas together with Theorems 5 and 9 and we obtain the following result:

Corollary 3 *Let Γ a finite non-basic set of boolean relations, then:*

a.- The set $\exists\text{-}\mathrm{Form}_C(\Gamma)$ contains $[x \vee y \vee \overline{z}]$ or $[\overline{x} \vee \overline{y} \vee z]$.

b.- The class $\mathcal{C}_{MBC}$ is pwm-reducible to $\mathcal{C}_{\exists\text{-}\mathrm{Form}_C(\Gamma)}$.

c.- The class $\mathcal{C}_{BC}$ is pwm-reducible to $\mathcal{C}_{\exists\text{-}\mathrm{Form}_C(\Gamma)}$.

d.- The class $\mathcal{C}_{\exists\text{-}\mathrm{Form}_C(\Gamma)}$ is not polynomially predictable with membership queries under the assumption that public key encryption systems secure against CC-attack exist.

Finally, Theorem 4 follows from Theorem 8 and Corollary 3. ■

Chapter 4

Extension to Constant-Free Formulas

4.1 Introduction

So far, there are only two complexity results about generalized quantified boolean formulas. The first one, due to Schaefer [81] is a dichotomy result for the satisfiability problem (see Theorem 3). The second one concerns the learnability of generalized quantified boolean formulas and it was presented in the previous chapter (see Theorem 4). It is remarkable that both results, as well as some other known results over generalized (quantifier-free) formulas, such as the inverse satisfiability problem [54], assume that constants $\{0, 1\}$ are available. This apparently simple restriction reduces considerably the range of cases that need to be considerated simplifying the proof. As we will see later (Chapter 6) studying the connection of complexity problems over quantified problems with clones in universal algebras, availability of constants implies that only idempotent clones need to be considerated.

In fact, Schaefer's original proof [81] for the satisfiability of generalized (quantifier-free) boolean formulas deals with formulas with constants as an intermediate step. Nevertheless, whereas for the quantifier-free case, the extension of results from $\text{Form}_C(\Gamma)$ to $\text{Form}(\Gamma)$ is rather straightforward, the equivalent problem for generalized quantified formulas is much more involved.

In this chapter we address the problem of extending the complexity results over generalized quantified boolean formulas with constants to constant-free formulas. First, we identify the conditions under which the availability of constants increases the expressivity power of formulas. We prove that, in a sense that will be precised later, class $\exists\forall\text{-}\operatorname{Form}_C(\Gamma)$ is strictly more expressive than $\exists\forall\text{-}\operatorname{Form}(\Gamma)$ if and only if

(a) Every relation in Γ is definable by a conjunction of clauses in which each clause is either a monotone disjunction of literals or the equality between two variables (Γ is equality monotone).

(b) Every relation in Γ is definable by a conjunction of clauses in which each clause is either a antimonotone disjunction of literals or the equality between two variables (Γ is equality antimonotone).

(c) Every relation in Γ is the set of solutions of an homogeneous system of linear equations over the two-element field $\{0, 1\}$ (Γ is 0-affine).

(d) Every relation in Γ is the set of solutions of a system of a linear equations over the two-element field $\{0, 1\}$ with particular solution $\langle 1, \ldots, 1\rangle$ (Γ is 1-affine).

This result is interesting because it is stated purely in terms of expressivity power and no complexity notions are involved. In consequence, it can be applied to extend different complexity results over quantified boolean formulas to constant-free formulas.

In particular we use the previous result to extend the abovementioned results about satisfiability and learnability of generalized quantified boolean formulas to constant-free formulas, completing the knowledge about the subjects. It is remarkable that, in both cases, hardness results carry through constant-free formulas. Let us state here, for the sake of completeness, the dichotomic classification for the satisfiability and learnability of quantified constant-free formulas. Given a set of boolean relation Γ, the satisfiability of $\exists\forall\text{-}\operatorname{Form}(\Gamma)$ is polynomial-time decidable if and only if the basis Γ is horn, antihorn, affine or bijunctive. Furthermore, the class $\exists\forall\text{-}\operatorname{Form}(\Gamma)$ is "polynomially learnable" if and only if the basis Γ is weakly monotone, weakly antimonotone, affine or bijunctive.

Thus, we can infer that availability of constants does not make any difference with respect to the satisfiability and learnability of quantified boolean formulas.

4.2 Main Result

The main question addressed in this chapter is: how the ability of placing boolean constants $\{0, 1\}$ in quantified formulas affects the expressivity power of quantified formulas? In particular, we are interested in how it affects the complexity of some computational problems over quantified formulas, such as satisfiability or learnability.

Thus we are interested in identifying for which classes of formulas the use of constants does matter. The same result, for quantification-free formulas was proven by Schaefer [81]. Before stating Schaefer's result, we introduce some definitions.

A boolean relation R of rank k is called *complementive* if for every assignment $t \in \{0, 1\}^k$, $t \in R$ if and only if $\neg t \in R$. This definition can be extended naturally to formulas, sets of boolean relations, and sets of formulas. The sets of complementive boolean relations satisfy another nice property: Let Γ be a complementive set of boolean relations, then $\exists\forall\text{-}\operatorname{Rel}(\Gamma)$ is also complementive.

Theorem 10 *[81] Let Γ be a set of boolean relations. Then at least one of the following conditions holds:*

a.- Every relation in Γ is 0-valid.

b.- Every relation in Γ is 1-valid.

c.- The boolean relations $[x]$ and $[\overline{x}]$ are contained in $\exists\text{-}\operatorname{Rel}(\Gamma)$.

d.- The boolean relation $[x \not\equiv y]$ is contained in $\exists\text{-}\operatorname{Rel}(\Gamma)$ and every relation in Γ is complementive.

Furthermore, if Γ satisfies at least one of conditions (a) or (b), then Γ satisfies none of conditions (c) or (d), and conversely, if Γ satisfies at least one of conditions (c) or (d), then Γ satisfies none of conditions (a) or (b).

Theorem 10 should be interpreted in the following way: Conditions (a) and (b) characterize the basis for which the availability of place constants adds extra power. If a basis Γ does not fall in any of the previous classes then Γ satisfies at least one of conditions (c) or (d) which express two different ways to simulate the use of constants. In the first case (condition (c)), take an arbitrary formula Φ with constants and replace every occurrence of constants 0 and 1 by the new variables y_0 and y_1. Finally simulate Φ with the construction $\exists y_0, y_1(\Phi \wedge \overline{y}_0 \wedge y_1)$. In the second case (condition d) we do the same replacement and we construct the new formula $\exists y_0, y_1(\Phi \wedge (y_0 \neq y_1))$. This formula is not exactly equivalent to Φ but it is equivalent "under dualization", which turns out to be sufficient for some computational problems, such as satisfiability and learnability.

The aim of this section is to state the counterpart of Theorem 10 for generalized quantified boolean formulas. The proof is given in Section 4.5.

Theorem 11 *Let Γ be a set of boolean relations. Then at least one of the following conditions holds:*

a.- Every relation in Γ is 0-affine.

b.- Every relation in Γ is 1-affine.

c.- Every relation in Γ is equality antimonotone.

d.- Every relation in Γ is equality monotone.

e.- The boolean relations $[x]$ and $[\overline{x}]$ are contained in $\exists\forall\text{-}\mathrm{Rel}(\Gamma)$.

f.- The boolean relation $[x \not\equiv y]$ is contained in $\exists\forall\text{-}\mathrm{Rel}(\Gamma)$ and every relation in Γ is complementive.

Furthermore, if Γ satisfies at least one of conditions (a), (b), (c), or (d), then, Γ satisfies none of conditions (e) or (f), and conversely, if Γ satisfies at least one of conditions (e) or (f), then Γ satisfies none of conditions (a), (b), (c), or (d).

4.3 Applications of the Main Result

Theorem 11 characterizes the cases in which the use of constants allows to increase the expressivity power. Now, we use the previous result to derive computational complexity results. Theorem 11 has two interesting consequences: first, it is stated in terms of expressivity power only; second, the classification is complete in the sense mentioned above. In consequence, Theorem 11 could be applied to establish when the availability of constants affects the complexity of diverse computational problems. We present here two examples: satisfiability and learnability of quantified boolean formulas.

4.3.1 Satisfiability of Quantified Boolean Formulas

We start by stating the dichotomy theorem for the satisfiability of constant-free quantified boolean formulas.

Theorem 12 *(Dichotomy Theorem for the satisfiability of (constant-free) quantified boolean formulas) Let* Γ *be a finite set of boolean relations. If* Γ *satisfies one of the conditions* (a)-(d) *below, then* $SAT(\exists\forall\text{-}\,\mathrm{Form}(\Gamma))$ *is polynomial-time decidable. Otherwise,* $SAT(\exists\forall\text{-}\,\mathrm{Form}(\Gamma))$ *is log-complete in PSPACE.*

a.- Γ *is bijunctive.*

b.- Γ *is affine.*

c.- Γ *is horn.*

d.- Γ *is antihorn.*

This result extends a dichotomy classification for the satisfiability of quantified boolean formulas with constants due to Schaefer (Theorem 3)

There are two remarkable facts to note here: First, when we consider boolean formulas with constants the tractable cases remain unchanged if we allow both universal and existential quantifiers, only existential quantifiers or no quantifiers at all. However, in the two last cases the intractable cases become NP-complete instead of PSPACE-complete.

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

Proof of Theorem 12

This proof follows the lines of the proof of Theorem 2.1 in [81]. Tractability of classes $(a)-(d)$ is an obvious consequence of the theorem for formulas with constants.

The PSPACE-hardness result follows from Theorem 11 taking into account that the equality antimonotone (resp. equality monotone) relations are a subset of horn relations (resp. antihorn relations) and that the 0-affine and 1-affine relations are a subset of affine relations. Therefore if conditions

$(a)-(d)$ in Theorem 12 do not hold, then conditions (e) or (f) in Theorem 11 hold. In both cases we can prove the following log-space reduction

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

Therefore, $\mathrm{SAT}(\exists\forall\text{-}\mathrm{Form}(\Gamma))$ is PSPACE-hard if so is the corresponding problem with constants.

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

In both cases Φ is satisfiable if and only Ψ is, and Ψ is logspace computable given Φ.

■

4.3.2 Learnability of Quantified Boolean Formulas

Theorem 11 can be applied to get rid of constants some other results over quantified boolean formulas. In this section, we complete a previous result on the learnability of quantified boolean formulas with constants proved in Chapter 3. Our dichotomy for the learnability of constant-free quantified boolean formulas is stated right now.

Theorem 13 *(Dichotomy Theorem for the learnability of (constant-free) quantified boolean formulas) Let Γ a finite set of boolean relations. If Γ satisfies one of the conditions (a)-(d) below, then $\mathcal{C}_{\exists\forall\text{-}\mathrm{Form}(\Gamma)}$ is polynomially exactly learnable with improper equivalence queries. Otherwise, $\mathcal{C}_{\exists\forall\text{-}\mathrm{Form}(\Gamma)}$ is not polynomially predictable with membership queries under the assumption that public key encryption systems secure against chosen ciphertext attack exist.*

a.- Γ is bijunctive.

b.- Γ is affine.

c.- Γ is weakly monotone.

d.- Γ is weakly antimonotone.

Proof. The learnable cases follow directly from Theorem 4, since the problem without constants becomes easier and therefore we can use the algorithm for learning formulas with constants to learn formulas without constants.

The non-learnability results follow from Theorem 11 taking into account that the equality antimonotone (resp. equality monotone) relations are a subset of weakly antimonotone (resp. weakly monotone relations) and that the 0-affine and 1-affine relations are a subset of affine relations. Therefore, if conditions $(a)-(d)$ in Theorem 4 do not hold, then conditions (e) or (f) in Theorem 11 hold. In both cases we can prove the following pwm-reduction

$$\mathcal{C}_{\exists\forall\text{- Form}_C(\Gamma)} \leq_{pwm} \mathcal{C}_{\exists\forall\text{- Form}(\Gamma)}.$$

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

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

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

4.4 Semantic Characterizations

In this section we give semantic characterizations of some sets of boolean relations. We present semantic characterizations for the 0-affine, antimonotone, and equality antimonotone boolean relations. Semantic characterizations for the dual classes are not stated but are used implicitly later. Other semantic characterizations can be found in Chapter 3.

The following result is known (see [17], for example). It characterizes the class of the 0-affine sets. It shows that 0-affine relations express vectorial subspaces over the two-element field $\{0, 1\}$.

Lemma 18 *[17](Semantic characterization of* 0*-affine relations) Let* R *be a boolean relation of rank* k*. Then the following statements are equivalent:*

a.- R *is* 0*-affine.*

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

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

At this point we show a characterization of the antimonotone and equality antimonotone relations. The key ingredient in the results of this section is the notion of k-compatibility, as defined in Section 3.2.1. Accordingly, the following result is obtained adapting slightly the proof of Lemma 6.

Lemma 19 *(Semantic characterization of antimonotone boolean relations) Let* R *be a boolean relation of rank* k*. Then the following statements are equivalent:*

a.- R *is antimonotone.*

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

c.- For every $T \subseteq \{1, \ldots, k\}$ *and for every assignment* t *not in* $\mathrm{pr}_T\, R$ *and* $|T|$*-compatible with* $\mathrm{pr}_T\, R$*, we have* $t = 1^{|T|}$*.*

Proof. Equivalence between conditions (a) and (b) is well known. Equivalence between conditions (a) and (c) is obtained adapting the proof of Lemma 6 ∎

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

Lemma 20 *(Semantic characterization of equality antimonotone boolean relations) Let R be a boolean relation of rank k. Then the following statements are equivalent:*

a.- R is equality antimonotone.

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

i.- For all $1 \leq i \neq j \leq l$, $T_i \cap T_j = \emptyset$ and $\bigcup_{i=1}^{l} T_i = \{1, \ldots, k\}$, and

ii.- For all $t \in R$, $1 \leq i \leq l$, and $j, m \in T_i$, $t[j] = t[m]$, and

iii.- The boolean formula $R(x_1, \ldots, x_k)\left[\begin{smallmatrix}T_1\\ y_1\end{smallmatrix}\right] \cdots \left[\begin{smallmatrix}T_l\\ y_l\end{smallmatrix}\right]$ is antimonotone.

Proof.
[$(\boldsymbol{a}) \Longrightarrow (\boldsymbol{b})$] There exists some formula over the variables $x_1, \ldots, x_k$, $\Phi = \bigwedge_{j=1}^{s} C_j$ where each C_j is an equality between two variables or a disjunction of negated variables such that $[\Phi] = R$. Let $\sim$ be a binary relation over $\{1, \ldots, k\}$ defined in the following way: $i \sim j$ iff $x_i = x_j$ is a clause in Φ, and let $\sim^*$ be its reflexive, symmetric, and transitive closure. Clearly $\sim^*$ is an equivalence relation and therefore defines a partition $T_1, \ldots, T_l$ over the set $\{1, \ldots, k\}$. It is easy to verify that this partition satisfies conditions (bi), (bii), and ($biii$).
[$(\boldsymbol{b}) \Longrightarrow (\boldsymbol{a})$] Let $\{T_1, \ldots, T_l\}$ be a collection of sets that satisfies condition (b). Let $\Phi = \wedge_{j=1}^{s} C_j$ be a boolean formula in the variables $y_1, \ldots, y_l$ where each clause is antimonotone and such that $[\Phi] = R(x_1, \ldots, x_l)\left[\begin{smallmatrix}T_1\\ y_1\end{smallmatrix}\right] \ldots \left[\begin{smallmatrix}T_l\\ y_l\end{smallmatrix}\right]$. For every $1 \leq i \leq l$, let $T_i = \{j_1^i, \ldots, j_{r_i}^i\}$ where $r_i = |T_i|$. We define Φ_i in the following way:

$$\Phi_i = (x_{j_1^i} = x_{j_2^i}) \wedge \cdots \wedge (x_{j_{r_i-1}^i} = x_{j_{r_i}^i})$$

Then we construct the following formula:

$$\Psi = \left(\Phi \begin{bmatrix} y_1 \\ x_{j_1^1} \end{bmatrix} \cdots \begin{bmatrix} y_l \\ x_{j_1^l} \end{bmatrix} \right) \wedge \bigwedge_{i=1}^{l} \Phi_i$$

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

■

4.5 Proof of the Main Result (Theorem 11)

It is easy to show that every one of conditions $(a) - (d)$ is preserved under conjunction, quantification, renaming and identification of variables. In consequence, if a basis Γ is equality monotone (resp. equality antimonotone, 0-affine, 1-affine) then every formula in $\exists\forall\text{-}\operatorname{Form}_C(\Gamma)$ is equality monotone (resp. equality antimonotone, 0-affine, 1-affine). Thus, the incompatibility between conditions $(a) - (d)$ and conditions $(e) - (f)$ follows from the fact that if any of the conditions (e) or (f) holds, then the basis Γ is neither 0-valid or 1-valid.

To prove that at least one of conditions $(a) - (f)$ is satisfied we use a case analysis. We start considering separately complementive and non-complementive sets of boolean relations. First, we will see that any non-complementive set of boolean relations can generate the constants, i.e, the relations $[x]$ and $[\overline{x}]$.

Non-Complementive relations

We start with a useful characterization of complementive relations.

Lemma 21 *A set Γ of boolean relations is not complementive if and only if $\{[x], [\overline{x}], [x \vee \overline{y}]\} \cap \exists\text{-}\operatorname{Rel}(\Gamma) \neq \emptyset$.*

Proof. The proof is one way is immediate. For the converse, consider a k-ary relation R in Γ and an assignment t in R such that $\neg t \notin R$. Consider now the relation

$$R(x_1, \ldots, x_k) \left[\begin{matrix} \{x_i : t[i] = 0\} \\ y_0 \end{matrix}, \begin{matrix} \{x_i : t[i] = 1\} \\ y_1 \end{matrix} \right],$$

obtained replacing every variable assigned to 0 in t by y_0 and every variable assigned to 1 in t by y_1. Clearly, the previous relation is equivalent to one of the following relations: $[\overline{y_0}]$, $[y_1]$, or $[\overline{y_0} \vee y_1]$. ∎

If we have $[\overline{x} \vee y]$ we are done, since we can obtain the constants by quantifying universally any of the variables, that is $(y) \equiv \forall x(\overline{x} \vee y)$ and $(\overline{x}) = \forall y(\overline{x} \vee y)$. Actually this is the only place dealing with non-complementive relations in which we use the universal quantification. This fact explains the nature of the difference between the quantified and non-quantified case with respect to the constants in the non-complementive case: with the universal quantification we can simulate both constants when the implication relation is available. Furthermore, this is the unique case in which universal quantification provides extra expressibility power.

Now, we continue with our case study. Along the rest of the section we will assume that relations are not complementive.

Let us assume that relation $[\overline{x}]$ is included in the basis. Thus we can assume that we can put the constant 0 arbitrarily. Since for any formula Φ, the formula $[\Phi\left[\begin{smallmatrix} x \\ 0 \end{smallmatrix}\right]]$ is identical to $[\exists x(\Phi \wedge \overline{x})]$.

Consider the case in which there exists a non-0-valid relation R in the basis. In consequence, there exists some assignment t containing at least one 1 in R. Consider the 1-ary relation obtained by assigning variable y to all the components assigned to 1 and enforcing the other components to be 0, that is,

$$[R(x_1, \ldots, x_k)\begin{bmatrix} \{x_i : t[i] = 0\} & \{x_i : t[i] = 1\} \\ 0 & y \end{bmatrix}] = [y].$$

Thus, we can assume that the basis Γ is 0-valid. Now consider the case that Γ is not 0-affine. We will see that in this case it is possible to generate the relation $[\overline{x} \vee \overline{y}]$.

Lemma 22 *Let R be a 0-valid non-0-affine boolean relation of rank k. Then $\{[\overline{x} \vee \overline{y}], [x \vee \overline{y}]\} \cap \exists\text{-}\mathrm{Rel}(\{R, [\overline{x}]\}) \neq \emptyset$.*

Proof.
Let t_1 and t_2 be assignments such that condition (bii) in Lemma 18 fails. Let $\Phi \in \exists\text{-}\mathrm{Rel}(\{R, [\overline{x}]\})$ be a boolean formula defined in the following way:

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

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

Note that $(0, 0, 0)$, $(1, 0, 1)$, and $(0, 1, 1)$ are in $[\Phi]$ but not $(1, 1, 0)$. If $(1, 1, 1) \notin [\Phi]$, then $(\exists z \Phi) \equiv (\overline{x} \vee \overline{y})$ and we are done. It is easy to see that every 0-valid, 1-valid, non-complementive relation R can generate the implication in the following way: Let t be the tuple proving the non-complementivity, that is $t \in [\Phi]$ and $\neg t \notin [\Phi]$. Then,

$$R(x_1, \ldots, x_k)\begin{bmatrix} \{x_i : t[i] = 0\} & \{x_i : t[i] = 1\} \\ y_0 & y_1 \end{bmatrix} \equiv (\overline{y_0} \vee y_1).$$

Thus, $[\Phi]$ contains $(0,1,0)$ and $(1,0,0)$ but not $(0,0,1)$. Now, we know the complete truth-table. We obtain the implication by

$$\Phi\begin{bmatrix} x \\ 0 \end{bmatrix} \equiv (y \vee \overline{z}).$$

■

Now, we are going a see that non-weakly antimonotone boolean relations can generate $[x]$ or $[\overline{x} \vee y]$, if relation $[\overline{x} \vee \overline{y}]$ is available. First, we prove it for non-horn relations (this result appears as a particular case in the general result).

Lemma 23 *Let R be a 0-valid non-horn boolean relation of rank k. Then $\{[x], [x \vee \overline{y}]\} \cap \exists\text{-}\mathrm{Rel}(\{R, [\overline{x} \vee \overline{y}]\}) \neq \emptyset$.*

Proof.
Let $T \subseteq \{1, \ldots, k\}$ be a subset of indices and t be a $|T|$-compatible assignment with $\mathrm{pr}_T\, R$ not in $\mathrm{pr}_T\, R$ that contains at least two zeros, say corresponding to indices i, j (The existence of such assignment is guaranteed by Lemma 7). Let $\Phi \in \exists\text{-}\mathrm{Form}(\{R\})$ be a boolean formula over the variables $\{x_1, \ldots, x_{m=|T|}\}$ equivalent to $\mathrm{pr}_T\, R$.

If $|t|_1 = 0$ then $\Phi\left[\begin{smallmatrix} \{x_l : l \neq i\} \\ 0 \end{smallmatrix}\right] \equiv (x_i)$ and we are done.

Otherwise, let $V = \{x_l : l \in T/\{i,j\} \;\; t[l] = 0\}$ be the set of the remaining variables with value zero. We construct the new formula

$$\Psi = \left(\Phi\begin{bmatrix} V & \{x_1, \ldots, x_m\} - V \\ 0 & y \end{bmatrix}\right) \wedge (\overline{x_i} \vee \overline{x_j}).$$

Formula Ψ is a boolean formula over the variables variables $\{y, x_i, x_j\}$. We know that $(0,0,0) \in [\Psi]$ (by the 0-validity) and $(1,0,0) \notin [\Psi]$, $(1,0,1) \in [\Psi]$, $(1,1,0) \in [\Psi]$ (by the compatibility). Furthermore $(0,1,1), (1,1,1) \notin [\Psi]$ (by the clause $(\overline{x_i} \vee \overline{x_j})$).

We can infer that $(0,0,1) \notin [\Psi]$. Otherwise $\Psi\left[\begin{smallmatrix} x_i \\ 0 \end{smallmatrix}\right] \equiv (\overline{y} \vee x_j)$. Similarly, we have $(0,1,0) \notin [\Psi]$. Thus, we have the complete truth-table of Ψ. We obtain $(\overline{y} \vee x_i)$ with the formula $(\exists x_j \Psi)$. ■

Lemma 24 *Let R be a 0-valid non-equality antimonotone boolean relation of rank k. Then $\{[x], [x \vee \overline{y}]\} \cap \exists\text{-}\mathrm{Rel}(\{R, [\overline{x} \vee \overline{y}]\}) \neq \emptyset$.*

Proof.

Let n be the arity of R. Consider the equivalence relation $\sim$ in $\{1,\ldots,n\}$ given by

$$i \sim j \iff \forall t \in R\; t[i] = t[j].$$

Let $T_1,\ldots,T_l$ the equivalence classes induced by the given equivalence relation. We construct the following formula:

$$\Psi(y_1,\ldots,y_l) = R(x_1,\ldots,x_n)\begin{bmatrix}\{x_i : i \in T_1\}\\ y_1\end{bmatrix}\cdots\begin{bmatrix}\{x_i : i \in T_l\}\\ y_l\end{bmatrix}$$

By Lemma 20, Ψ is not antimonotone. Let T be a subset and t an assignment such that condition (c) in Lemma 19 fails. Let Φ be a boolean formula over the variables $\{x_1,\ldots,x_{|T|}\}$ equivalent to $\mathrm{pr}_T[\Psi]$. We can suppose that t contains only one zero, otherwise Φ is not horn and we can apply Lemma 23. We study three cases:

- $|T| = 1$. In this case $t = 0$ and $[\Phi] = [x]$
- $|T| = 2$. Suppose that $t = (1,0)$. In this case $(0,0) \in [\Phi]$ (by the 0-validity) and $(1,1) \in [\Phi]$ (by the $|T|$-compatibility). We know that $(0,1) \in [\Phi]$, otherwise both indices would belong to the same equivalence class. Thus $[\Phi] = [\overline{x} \vee y]$.
- $|T| \geq 3$. This case is handled similarly to Lemma 12. Recall that for every assignment t, every index $j \leq |t|$, and every constant $b \in \{0,1\}$, $t_{j\leftarrow b}$ as the assignment obtained from t replacing the jth element by b.

  Let i be the index corresponding to the value 0 in t. By the $|T|$-compatibility we know that $t_{0\leftarrow j} \in [\Phi]$ when $1 \leq j \neq i \leq |T|$. Therefore for every $t' \neq t$ such that $t'[i] = 0$, we have $t' \in [\Phi]$. Otherwise Φ is not horn, since $t' = \bigwedge_{j\neq i, t'[j]=0} t_{0\leftarrow j}$. By the $|T|$-compatibility $1^{|T|} \in [\Phi]$. Clearly $\neg t \notin R$, otherwise $\Phi\left[\begin{smallmatrix}\{x_m : m\in T, t[m]=1\}\\ y\end{smallmatrix}\right] \equiv (\overline{y} \vee x_i)$. Now, pick any arbitrary index $j \neq i$ and consider the assignment t'' given by: $t''[m] = 1$ if $(m = i)$ or $(m = j)$ and $t''[m] = 0$ otherwise. If $t'' \in [\Phi]$ then we have $\Phi\left[\begin{smallmatrix}\{x_m : m\in T/\{i,j\}\}\\ 0\end{smallmatrix}\right] \equiv (x_j \vee \overline{x_i})$. Otherwise, $\Phi\left[\begin{smallmatrix}\{x_i,x_j\}\\ x\end{smallmatrix}, \begin{smallmatrix}\{x_m : m\in T/\{i,j\}\}\\ y\end{smallmatrix}\right] \equiv (\overline{x} \vee y)$

$\blacksquare$

Thus, summarizing we have

Lemma 25 *Let Γ be a set of non-0-affine, non-equality antimonotone boolean relations. Then* $\exists\text{-}\operatorname{Rel}(\Gamma \cup \{[\overline{x}]\}) \cap \{[x], [x \vee \overline{y}]\} \neq \emptyset$.

Similarly,

Lemma 26 *Let Γ be a set of non-1-affine, non-equality monotone boolean relations. Then* $\exists\text{-}\operatorname{Rel}(\Gamma \cup \{[x]\}) \cap \{[x], [x \vee \overline{y}]\} \neq \emptyset$.

Putting all together, we get the classification for non-complementive basis.

Corollary 4 *Let Γ be a non-complementive set of boolean relations. If Γ is not 0-affine, not 1-affine, not equality monotone and not equality antimonotone then* $\{[x], [\overline{x}]\} \subseteq \exists\forall\text{-}\operatorname{Rel}(\Gamma)$.

Complementive relations

Complementive relations are easier. We have to prove that the inequality relation $[x \neq y]$ can be generated. If a k-ary relation R is not 0-valid (and consequently not 1-valid) then consider any arbitrary assignment $t \in R$. Then the formula $R(x_1, \ldots, x_k)\left[\begin{smallmatrix}\{i:t[i]=1\} \\ x\end{smallmatrix}, \begin{smallmatrix}\{i:t[i]=0\} \\ y\end{smallmatrix}\right] \equiv (x \neq y)$ and we are done.

For 0-valid relations the analysis is more involved. Basically, we have to identify the cases in which a relation losses its 0-validity under universal quantification. Here, we present an example.

Lemma 27 *Let R be a boolean relation of rank k. If there exist two different assignments $t, t' \in R$ such that $t \leq t'$ and $t \oplus t' \notin R$, then $\exists\forall\text{-}\operatorname{Rel}(\{R\})$ is not 0-valid.*

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

We will see that any complementive non-0-affine relation R can generate a non-0-valid relation. In consequence, since the complementivity is preserved, we reduced the problem to the previous case in which relation $[x \neq y]$ can be generated.

Lemma 28 *Let R be a complementive non-0-affine boolean relation of rank k. Then the class $\exists\forall\text{-}\mathrm{Rel}(\{R\})$ is not 0-valid.*

Proof. Let $t_1, t_2 \in R$ be assignments such that condition (bii) in Lemma 18 is falsified. Let $V_{i,j}$ $(0 \leq i, j \leq 1)$ disjoint sets of variables defined by $V_{i,j} = \{x_l : 1 \leq l \leq k : t_1[l] = i \wedge t_2[l] = j\}$. Let us assume that sets $T_{i,j}$ $0 \leq i \leq j$ are not empty (the remaining cases are handled easily). We define the following formula

$$\Phi = R(x_1, \ldots, x_k) \begin{bmatrix} \frac{V_{0,0}}{x_{0,0}}, \frac{V_{0,1}}{x_{0,1}}, \frac{V_{1,0}}{x_{1,0}}, \frac{V_{1,1}}{x_{1,1}} \end{bmatrix}$$

Let us study $[\Phi]$. Assignments $(0,0,0,0)$, $(0,1,0,1)$, $(0,0,1,1)$ are in $[\Phi]$ and assignment $(0,1,1,0)$ is not in $[\Phi]$.

We can infer $(0,0,1,0) \in [\Phi]$. Otherwise, formula $(\exists x_{0,1}\Phi)$ is satisfied by $t'_1 = (0,0,1)$ and $t'_2 = (0,1,1)$ but not by $t'_1 \oplus t'_2 = (0,1,0)$. Similarly, we can infer $(0,1,0,0) \in [\Phi]$. Since R is complementive, $(1,1,0,1) \in [\Phi]$ and $(1,0,0,1) \notin [\Phi]$. Now, the previous lemma can be applied taking $(0,1,0,0)$ and $(1,1,0,1)$ in order to infer that $\exists\forall\text{-}\mathrm{Rel}(\{R\})$ is not 0-valid. ■

Thus, we have

Lemma 29 *Let Γ be a complementive non-0-affine set of boolean relations. Then $[x \neq y] \in \exists\forall\text{-}\mathrm{Rel}(\Gamma)$.*

The proof is complete. If basis Γ is non-complementive, then $[x], [\overline{x}] \in \exists\forall\text{-}\mathrm{Rel}(\Gamma)$ and condition (e) is satisfied. Otherwise, $[x \neq y] \in \exists\forall\text{-}\mathrm{Rel}(\Gamma)$ and condition (f) is satisfied.

Chapter 5

Algebra Universal and Clones

5.1 Introduction

In the previous chapter we studied some computational problems over generalized boolean formulas. From some chapters on, we will be interested in studying problems over larger domains. As a consequence of the combinatorial explosion typical of larger domains, the "elementary techniques" used in the boolean case are not applicable any more. In this chapter we introduce a new set of tools used intensively in the study of quantified formulas over non-boolean domains. The main idea of our approach is to exploit a link between generalized quantified formulas and the notion of *clone* studied in Universal Algebra. The use of Clone Theory in complexity theory was introduced by Jeavons et al. in [51] and it has been very fruitful in the study of Constraint Satisfaction Problems (see Chapter 6).

This chapter is devoted to introducing the concept of clone and some other algebraic concepts that will be of intensive use in next chapters. No complexity notions are involved. The material presented in this chapter is well known with the exception of two new results: the proof of the Galois connection (Theorem 15) between clones and co-clones in the finite case, which is new and simplifies the original proof in [36], and the family of coset generating operations and its properties. Coset generating operations are the natural extension of affine operations to non-necessarily abelian groups.

Let us introduce very briefly the notion of clone. A clone is a set F of operations that satisfies the following conditions: (1) F contains the projections, ie., operations that return one of its arguments, and (2) F is closed under composition.

The importance of clones relies in the following fact: Every set of relations Γ on D has naturally attached a clone of operations $\mathrm{Pol}(\Gamma)$, called clone of polymorphims of Γ. As we will see in next chapters, the clone $\mathrm{Pol}(\Gamma)$ approximately characterizes the expressivity power of the set of generalized quantified formulas over basis Γ. We say "approximately" because some technical assumptions are needed to have a complete equivalence. Nevertheless, we will see that when we move into computational problems, such as satisfiability or learnability, those technical difficulties can be overcome easily. In consequence, we get the following interesting result: The complexity of the satisfiability problems and the learnability problem over generalized quantified formulas over a basis Γ depends only on is clone of polymorphisms $\mathrm{Pol}(\Gamma)$. We assume familiarity with algebraic concepts (see [64, 12, 72, 84, 75] for further details)

5.2 Some Algebraic Terminology

Definition 4 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 5 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 $\mathbf{A}$.

Now consider similar algebras $\mathbf{A}$ and $\mathbf{B}$ and let ϱ be an operation symbol of rank r. A function h from A into B is said to *respect the interpretation of* ϱ if and only if

$$h(\varrho^{\mathbf{A}}(a_1, \ldots, a_r)) = \varrho^{\mathbf{B}}(h(a_1), \ldots, h(a_r))$$

for all $a_1, \ldots, a_r \in A$.

Definition 6 Let $\mathbf{A}$ and $\mathbf{B}$ be similar algebras. A function h from A into B is called a *homomorphism* from $\mathbf{A}$ into $\mathbf{B}$ if and only if h respects the interpretation of every operation symbol of $\mathbf{A}$. If h is bijective then we say that h is an *isomorphism*. In this case we say that $\mathbf{A}$ and $\mathbf{B}$ are isomorphic and we denote it by $\mathbf{A} \approx \mathbf{B}$.

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:

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

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 a n-ary relation over A. We say that f *preserves* R (R *is closed under* f, or f *is a polymorphism of* R) if

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

For arbitrary mapping $g : A^n \to A$, we define a subset $g^\square$ of A^{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 A, 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.

5.3 Closure Systems

The material presented in this section is fairly standard. See [12, 72] for further details. Let A be any set and $\mathcal{P}(A)$ be the set of all its subsets. We wish to consider certain subsets of $\mathcal{P}(A)$, or as we shall say, *systems* of subsets of A. A system $\mathcal{C}$ of subsets of A is said to be a *closure system* if $\mathcal{C}$ is closed under intersections, i.e.

$$\text{for any subsystem } \mathcal{D} \subseteq \mathcal{C}, \ \bigcap \mathcal{D} \in \mathcal{C}$$

Since a closure system admits arbitrary intersections, it follows that it is a complete lattice (with respect to the ordering by inclusion).

A second notion which we require (and which will turn out to be equivalent to that of a closure system) is that of a closure operator on a set. A *closure operator* on a set A is a mapping J of $\mathcal{P}(A)$ into itself with the properties:

1. If $X \subseteq Y$, then $J(X) \subseteq J(Y)$,
2. $X \subseteq J(X)$,

3. $J(J(X)) = J(X)$,

for all $X, Y \in \mathcal{P}(A)$.

Closure systems and closure operators are different mathematical embodiments of the same concept.

Theorem 14 *[89] Every closure system $\mathcal{C}$ on a set A defines a closure operator J on A by the rule*

$$J(X) = \bigcap\{Y \in \mathcal{C} : X \subseteq Y\}.$$

Conversely, every closure operator J on A defines a closure system by

$$\mathcal{C} = \{X \subseteq A : J(X) = X\},$$

and the correspondence between closure systems and closure operators thus defined is bijective.

We now give an example of closure systems, which will be of importance in what follows.

Let A and B be any sets and ϕ be a correspondence from A to B, i.e., a subset of $A \times B$. For any subset X of A we define a subset $v(X)$ of B by the equation

$$v(X) = \{y \in B : \langle x, y \rangle \in \phi \ \ \forall x \in X\},$$

and similarly, for any subset Y of B, we define a subset $w(Y)$ of A by

$$w(Y) = \{x \in A : \langle x, y \rangle \in \phi \ \ \forall y \in Y\}.$$

Thus we have the mappings v, w of $\mathcal{P}(A), \mathcal{P}(B)$ into each other, with the properties

1a. If $X_1 \subseteq X_2$, then $v(X_1) \supseteq v(X_2)$,

1b. If $Y_1 \subseteq Y_2$, then $w(Y_1) \supseteq w(Y_2)$,

2a. $X \subseteq (v \circ w)(X)$,

2b. $Y \subseteq (w \circ v)(Y)$,

3a. $(v \circ w \circ v)(X) = v(X)$,

3b. $(w \circ v \circ w)(Y) = w(Y)$.

Conditions (1) and (2) follow immediately from the definitions. It is easy to see that condition (3) follows from (1) and (2).

As a direct consequence of the previous properties, mappings $(w \circ v) : \mathcal{P}(A) \to \mathcal{P}(A)$ and $(v \circ w) : \mathcal{P}(B) \to \mathcal{P}(B)$ define closure systems.

A pair of mappings between $\mathcal{P}(A)$ and $\mathcal{P}(B)$, or more generally between any partially ordered sets, is called a *Galois connection* if it satisfies (1), (2) (and hence (3)). Most Galois connections encountered in practice arise from a correspondence between sets in the way described above.

5.4 Operations and Relations

The Galois connection that arises from the binary relation "preserves" (Definition 7) has been well studied (see [72] for example).

Definition 8 Let ϕ_A be the correspondence from the set of operations $\mathcal{F}^A$ on A to the set of relations $\mathcal{R}^A$ on A given by

$$\phi_A = \{(f, R) : f \in \mathcal{F}^A, R \in \mathcal{R}^A, f \text{ preserves } R\}.$$

The mappins v and w induced by the correspondence ϕ_A are called Pol_A (polymorphisms) and Inv_A (invariants) respectively. This mappings will play a central role in the next chapters. So they deserve a definition:

Definition 9 Let A be a finite set. We define mappings $\mathrm{Pol}_A : \mathcal{R}^A \to \mathcal{F}^A$, and $\mathrm{Inv}_A : \mathcal{F}^A \to \mathcal{R}^A$ as follows:

- For all $\Gamma \subseteq \mathcal{R}^A$ set of relations on A,

$$\mathrm{Pol}_A(\Gamma) = \{f \in \mathcal{F}^A : f \text{ preserves } R, \forall R \in \Gamma\}$$

- For all $F \subseteq \mathcal{F}^A$ set of operations on A,

$$\mathrm{Inv}_A(F) = \{R \in \mathcal{R}^A : f \text{ preserves } R, \forall f \in F\}$$

If the domain A is clear, we will refer Pol_A and Inv_A as Pol and Inv. Let us study these mappings. First we introduce the notion of clones and co-clones that will play a major role in the study of the Galois connection induced by ϕ_A.

5.4.1 Clones

Let $\mathcal{F}_n^A$ denote the set of all n-adic operations on A. Then, $\mathcal{F}^A = \bigcup_{n\geq 1} \mathcal{F}_n^A$.

Let $F \subseteq \mathcal{F}^A$ be a class of operations on A. We set $F_n = F \cap \mathcal{F}_n^A$ (the n-adic operations in F). We shall say that F is a *clone* if if satisfies the following conditions:

C_1 For each $1 \leq m \leq n$, F contains the *projection operation* $\mathrm{proj}_{n,m}$, defined by

$$\mathrm{proj}_{n,m}(x_1, \ldots, x_n) = x_m$$

C_2 For each $1 \leq n, m$, each $f \in F_n$ and each $g_1, \ldots, g_n \in F_m$. F_m contains the composite operation $h = f[g_1, \ldots, g_n]$ defined by

$$h(x_1, \ldots, x_m) = f(g_1(x_1, \ldots, x_m), \ldots, g_n(x_1, \ldots, x_m))$$

As a direct corollary of these conditions, we obtain some other simple conditions also satisfied by clones.

C_3 For every $f \in F_n$, the *cylindrification* $\mathrm{cyl}(f) \in \mathcal{F}_{n+1}^A$, defined by

$$\mathrm{cyl}(f)(x_1, \ldots, x_n, x_{n+1}) = f(x_1, \ldots, x_{n-1}, x_n)$$

belongs to F.

C_4 For every $f \in F_n$ $(n > 1)$, the *diagonalization* $\mathrm{diag}(f) \in \mathcal{F}_{n-1}^A$ defined by

$$\mathrm{diag}(f)(x_1, \ldots, x_{n-1}) = f(x_1, \ldots, x_{n-1}, x_{n-1})$$

belongs to F.

C_5 For every $f \in F_n$, the *rotation* $\mathrm{rot} \in \mathcal{F}_n^A$, defined by

$$\mathrm{rot}(f)(x_1, x_2, \ldots, x_{n-1}, x_n) = f(x_2, x_3, \ldots, x_n, x_1)$$

belongs to F.

C_6 For every $f \in F_n$, the *transposition* $\mathrm{trans}(f) \in \mathcal{F}_n^A$, defined by

$$\mathrm{trans}(x_1, x_2, \ldots, x_{n-1}, x_n) = f(x_1, x_2, \ldots, x_n, x_{n-1})$$

belongs to F.

C_7 For every $f \in F_n$ and $g \in F_m$, their *composition* $\mathrm{comp}(f,g) \in \mathcal{F}^A_{n+m-1}$, defined by

$$\mathrm{comp}(f,g)(x_1,\ldots,x_{n+m-1}) = f(x_1,\ldots,x_{n-1},g(x_n,\ldots,x_{n+m-1}))$$

belongs to F.

It is easy to verify that the clones over A constitute a closure system. There is a smallest clone which is the intersection of all clones; we shall denote it $\mathcal{I}_A$. It is easy to see that $\mathcal{I}_A$ contains exactly the projections over A. If $F \subseteq \mathcal{F}^A$ is any set of operations over A, there is a smallest clone containing all of the operations in F; this is the clone generated by F, and we denote it $\langle F \rangle_o$ (the subindex o denotes *operational*). If $F = \{f_1,\ldots,f_k\}$ is a finite set, we may write $\langle f_1,\ldots,f_k \rangle_o$ for $\langle F \rangle_o$, and refer to $f_1,\ldots,f_k$ as generators of $\langle F \rangle_o$.

Definition 10 The *clone of term operations* of an algebra $\mathbf{A} = (A,F)$, denoted by $T(\mathbf{A})$, is the smallest clone containing the basic operations of $\mathbf{A}$, that is $T(\mathbf{A}) = \langle F \rangle_o$. A *term operation* of $\mathbf{A}$ is any operation in $T(\mathbf{A})$. Two algebras with a common universe are called *term equivalent* iff they have the same term operations.

An operation f on A is *idempotent* iff it satisfies the identity $f(x,\ldots,x) = x$. A clone on A is called idempotent iff it consists of idempotent operations, while an algebra (A,F) is idempotent iff its basic operations (or equivalently, its term operations) are idempotent. 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 $\mathbf{A_1}$.

5.4.2 Co-Clones

Let $\mathcal{R}^A_n$ denote the set of all n-ary relations on A. Then, $\mathcal{R}^A = \bigcup_{n\geq 1} \mathcal{R}^A_n$.

Let $\Gamma \subseteq \mathcal{R}^A$ be a class of relations on A. We set $\Gamma_n = \Gamma \cap \mathcal{R}^A_n$ (the n-adic relations in Γ). We shall say that Γ is a *co-clone* if if satisfies the following conditions:

CC_1 Γ contains the two trivial equivalence relations, namely eq_A and A^2 given by

$$\mathrm{eq}_A = \{\langle x,x \rangle : x \in A\}$$

$$A^2 = \{\langle x, y\rangle : x, y \in A\}$$

CC$_2$ For any $R \in \Gamma_n$, $S \in \Gamma_m$, the cartesian product $R \times S \in \mathcal{R}^A_{n+m}$ given by

$$R \times S = \{\langle x_1, \ldots, x_n, y_1, \ldots, y_m\}\rangle : \langle x_1, \ldots, x_n\rangle \in R, \langle y_1, \ldots, y_m\rangle \in S\},$$

belongs to Γ.

CC$_3$ For any $R \in \Gamma_n$ and any list of (not necessarily different) indices $I = (i_1, \ldots, i_m)$ over $\{1, \ldots n\}$, the projection $\mathrm{pr}_I\, R \in \mathcal{R}^A_m$ given by

$$\mathrm{pr}_I\, R = \{\langle x_{i_1}, \ldots, x_{i_m}\rangle : \langle x_1, \ldots, x_n\rangle \in R\}$$

belongs to Γ. A restricted projection is a projection in which the list of indices does not contain repeated elements. Sometimes, if confusion with some other subindex is not possible, we will denote the projection by R_I. Notice that the notion of projection defined here differs slightly from the one used in Chapters 3 and 4 in the boolean case. More precisaly, the definition here differs from the fact that the parameter of the projection is a *list* of indices instead of a *set* of indices.

CC$_4$ For any $R \in \Gamma_n$ $(n \geq 2)$ and any pair of indices $1 \leq i, j \leq n$, the identification $R_{i=j} \in \mathcal{R}^A_n$ given by

$$R_{i=j} = \{\langle x_1, \ldots, x_n\rangle : \langle x_1, \ldots, x_n\rangle \in R, x_i = x_j\}$$

belongs to Γ.

As a direct corollary of these conditions, we obtain some other simple conditions also satisfied by clones.

CC$_5$ For every $R \in \Gamma_n$, the *cylindrification* $\mathrm{cyl}(R) \in \mathcal{R}^A_{n+1}$, defined by

$$\mathrm{cyl}(R) = \{\langle x_1, \ldots, x_n, x_{n+1}\rangle : \langle x_1, \ldots, x_n\rangle \in R\}$$

belongs to Γ.

CC$_6$ For every $R \in \Gamma_n$, the *rotation* $\mathrm{rot}(R) \in \mathcal{R}^A_n$, defined by

$$\mathrm{rot}(R) = \{\langle x_2, x_3, \ldots, x_n, x_1\rangle : \langle x_1, x_2, \ldots, x_{n-1}, x_n\rangle \in R\}$$

belongs to Γ.

CC$_7$ For every $R \in \Gamma_n$ $(n > 1)$, the *transposition* $\mathrm{trans}(R) \in \mathcal{R}_n^A$, defined by

$$\mathrm{trans}(R) = \{\langle x_1, \ldots, x_n, x_{n-1}\rangle : \langle x_1, \ldots, x_{n-1}, x_n\rangle \in R\}$$

belongs to Γ.

CC$_8$ For every $R \in \Gamma_n$ $(n > 0)$, the *projection* $\mathrm{proj}(R) \in \mathcal{R}_{n-1}^A$, defined by

$$\mathrm{proj}(R) = \{\langle x_1, \ldots, x_{n-1}\rangle : \langle x_1, \ldots, x_n\rangle \in R\}$$

belongs to Γ.

CC$_9$ For every $R, S \in \Gamma_n$, the *intersection* $R \cap S \in \mathcal{R}_n^A$ belongs to Γ.

Clearly, the co-clones over A constitute a closure system. There is a smallest co-clone which is the intersection of all co-clones; we shall denote it $\mathcal{J}_A$. It comprises just those relations obtained from A^n $(n \geq 1)$ by imposing any number of equality conditions $x_i = x_j$ $(1 \leq i, j \leq m)$ on the coordinates. If $\Gamma \subseteq \mathcal{R}^A$ is any set of relations on A, there is a smallest co-clone containing all of the relations in Γ; this is the co-clone generated by Γ, and we denote it $\langle\Gamma\rangle_r$ (subindex r denotes *relational*). If $\Gamma = \{R_1, \ldots, R_k\}$ is a finite set, we may write $\langle R_1, \ldots, R_k\rangle_r$ for $\langle\Gamma\rangle_r$, and refer to $R_1, \ldots, R_k$ as generators of $\langle\Gamma\rangle_r$.

If in the previous definition we remove "eq_A" in condition CC$_1$, and we replace "projection" by "restricted projection" in condition CC$_3$ we obtain a related concept called *restricted co-clone*. That is, we shall say that $\Gamma \subseteq \mathcal{R}^A$ is a restricted co-clone if contains A and it is closed under cartesian product, restricted projection and identification. Accordingly, we denote by $\langle\Gamma\rangle_{re}$ the restricted co-clone generated by Γ.

5.5 The Galois Connection

Clones and co-clones characterize precisely the invariants defined by the Galois connection established by ϕ_A (Definition 8).

The following theorem was proved independently by Geiger [36] and Bodnarchuk et al. [7]. We present a proof that differs in some aspect from the previous ones.

Theorem 15 *[36, 7] Let A be a finite set. Let* Pol *and* Inv *be the mappings of $\mathcal{P}(\mathcal{F}^A)$, $\mathcal{P}(\mathcal{R}^A)$ into each other induced by the correspondence ϕ_A. Then* *a. For every set of operations* $F \in \mathcal{F}^A$*, we have* $\mathrm{Pol}(\mathrm{Inv}(F)) = \langle F\rangle_o$.

b. For every set of relations $\Gamma \in \mathcal{R}^A$*, we have* $\mathrm{Inv}(\mathrm{Pol}(\Gamma)) = \langle \Gamma\rangle_r$.

Proof. The proof depends on several results
$(\boldsymbol{i})$**[Pol(Inv(*F*))** $\supseteq$ $\langle \boldsymbol{F}\rangle_o$**].** This part is verified by checking that conditions C_1 and C_2 are satisfied by $\mathrm{Pol}(\mathrm{Inv}(F))$. More specifically, it is enough to prove that: (1) every relation is preserved by every projection and (2) the composition of polymorphisms is also a polymorphism.
$(\boldsymbol{ii})$**[Pol(Inv(*F*))** $\subseteq$ $\langle \boldsymbol{F}\rangle_o$**].**

We prove that for any operation g not in $\langle F\rangle_o$ there exists some relation $R \in \mathrm{Inv}(F)$ such that g does not preserve R.

Let G be a set of operations of the same arity n. Each member of G is a mapping $A^n \to A$, and hence, it can be codified by a tuple of length $p = |A|^n$ over A, provided we fix some bijection $\pi : A^n \to \{1, \ldots, p\}$ (for example the position in the lexicographical order). More precisely, for every $f \in G$ we define the p-ary tuple t_f as

$$t_f[\pi(a_1, \ldots, a_n)] = f(a_1, \ldots, a_n), \quad a_1, \ldots, a_n \in A$$

The key point of this codification is that for all (not necessarily different) $f_1, \ldots, f_m \in G$ and any m-ary operation g, we have $g^n(t_{f_1}, \ldots, t_{f_m}) = t_{g[f_1,\ldots,f_m]}$. Recall that g^n is the operation obtained applying g componentwise. Along this proof we will make some notation abuse and for any $t_1, \ldots, t_m$ n-ary tuples and any m-ary operation g, we will use $g(t_1, \ldots, t_m)$ to refer $g^n(t_1, \ldots, t_m)$.

We define $\mathcal{R}_G = \{t_f : f \in G\}$ as the p-ary relation containing the codification of the operations in G.

Suppose that g is n-adic, and let $\langle F\rangle_o \cap \mathcal{F}_n^A$ be the n-adic operations in $\langle F\rangle_o$. Let $R = \mathcal{R}_{\langle F\rangle_o \cap \mathcal{F}_n^A}$

First, we claim that g does not preserve relation R. The relation R contains n tuples $t_{\mathrm{proj}_{n,1}}, \ldots, t_{\mathrm{proj}_{n,n}}$ corresponding to the projection operations $\mathrm{proj}_{n,1}, \ldots, \mathrm{proj}_{n,n}$. Notice that $g(t_{\mathrm{proj}_{n,1}}, \ldots, t_{\mathrm{proj}_{n,n}})$ is precisely $t_g \notin R$. Thus, g does not preserve R.

It remains to be shown that $R \in \mathrm{Inv}(F)$. Let f be any operation in F. Suppose that f is m-adic, and let $t_{h_1}, \ldots, t_{h_m}$ tuples in R corresponding to operations $h_1, \ldots, h_m \in \langle F\rangle_o \cap \mathcal{F}_n^A$. Since $\langle F\rangle_o$ is a clone, we have $f[h_1, \ldots, h_m] \in \langle F\rangle_o$. But $f(t_{h_1}, \ldots, t_{h_m}) = t_{f[h_1,\ldots,h_m]}$, and so $f(t_{h_1}, \ldots, t_{h_m}) \in R$. Thus $R \in \mathrm{Inv}(F)$.

***(iii)*[Inv(Pol(Γ)) $\supseteq \langle\Gamma\rangle_r$].** This part is immediate from the definitions. Similarly to item (i), it is verified by checking conditions CC_1 through CC_4.
***(iv)*[Inv(Pol(Γ)) $\subseteq \langle\Gamma\rangle_r$].** We present here an alternative proof of this part that differs, up to our knowledge, from the others that appear in the literature. We start proving that for all $n > 0$, $\mathcal{R}_{\mathrm{Pol}(\Gamma)\cap\mathcal{F}_n^A} \in \langle\Gamma\rangle_r$.

Let R be any arbitrary relation in Γ and let $t_1, \ldots, t_n$ be arbitrary (not necessarily different) tuples in R. We define $R_{t_1,\ldots,t_n}$ as the relation associated to the set of operations that satisfy the preserving condition taking R and $t_1, \ldots, t_n$. That is

$$R_{t_1,\ldots,t_n} = \{t_f : f \in \mathcal{F}_n^A \;\; f(t_1, \ldots, t_n) \in R\}.$$

Notice that $R_{t_1,\ldots,t_n}$ can be constructed in the following way:

$$R_{t_1,\ldots,t_n} = \{t \in A^p : \langle t[\pi(t_1[1], \ldots, t_n[1])], \ldots, t[\pi(t_1[k], \ldots, t_n[k])]\rangle \in R\},$$

which implies that $R_{t_1,\ldots,t_m}$ can be constructed from R with a sequence of cylindrifiacions, in order to add the inessential indices, and with a sequence of rotations and transpositions in order to constraint the appropiate indices. Thus $R_{t_1\ldots,t_m} \in \langle R\rangle_r$

Finally, to obtain the n-ary operations preserving Γ, we enforce the previous property to all n-sets of tuples in all relations in Γ. We obtain $\mathcal{R}_{\mathrm{Pol}(\Gamma)\cap\mathcal{F}_n^A}$ as the conjunction:

$$\mathcal{R}_{\mathrm{Pol}(\Gamma)\cap\mathcal{F}_n^A} = \bigcap_{R\in\Gamma, t_1,\ldots,t_n\in R} R_{t_1,\ldots,t_m}$$

In consequence $\mathcal{R}_{\mathrm{Pol}(\Gamma)\cap\mathcal{F}_n^A} \in \langle\Gamma\rangle_r$.

The second step is to prove that if R is an k-ary relation in Inv(Pol(Γ)) with exactly n different tuples, then $R \in \langle\mathcal{R}_{\mathrm{Pol}(\Gamma)\cap\mathcal{F}_n^A}\rangle_r$. Let $t_1, \ldots, t_n$ be the tuples in R. For every index i ($1 \le i \le k$) we define $j_i = \pi(t_1[i], \ldots, t_n[i])$. Finally define T as

$$T = \mathrm{pr}_{j_1,\ldots,j_k}(\mathcal{R}_{\mathrm{Pol}(\Gamma)\cap\mathcal{F}_n^A})$$

$\mathcal{R}_{\mathrm{Pol}(\Gamma)\cap\mathcal{F}_n^A}$ contains the tuples associated to the projection operations: $t_{\mathrm{proj}_{n,1}}, \ldots, t_{\mathrm{proj}_{n,n}}$. Notice that $\mathrm{pr}_{j_1,\ldots,j_k} t_{\mathrm{proj}_{n,i}} = t_i$, thus we have $R \subseteq T$. Conversely, consider any tuple t_h in T corresponding to a n-adic operation h in Pol(Γ). Then we have $\mathrm{pr}_{j_1,\ldots,j_k} t_h = \mathrm{pr}_{j_1,\ldots,j_k} h(t_{\mathrm{proj}_{n,1}}, \ldots, t_{\mathrm{proj}_{n,n}}) = h(\mathrm{pr}_{j_1,\ldots,j_k} t_{\mathrm{proj}_{n,1}}, \ldots, \mathrm{pr}_{j_1,\ldots,j_k} t_{\mathrm{proj}_{n,n}}) = h(t_1, \ldots, t_n)$ which belongs to R since $h \in \mathrm{Pol}(\Gamma)$. So, $T \subseteq R$, and in consequence $R = T$.

Remark: We say that R is *canonical* if for all pairs of different indices $1 \leq i \neq j \leq k$ there exists some t_l $(1 \leq l \leq n)$ such that $t_l[i] \neq t_l[j]$. Notice that if R is canonical, then $j_1, \ldots, j_k$ does not contain repeated elements and in consequence, $\mathrm{pr}_{j_1,\ldots,j_l}(\mathcal{R}_{\mathrm{Pol}(\Gamma)\cap\mathcal{F}_n^A})$ is a restricted projection. Furthermore, notice that $(\mathcal{R}_{\mathrm{Pol}(\Gamma)\cap\mathcal{F}_n^A})$ is constructed using only cartesian product, relation A (in order to add the inessential variables), cylindrification, rotation, and transposition. All this operations can be simulated using relation A, cartesian product, identification, and restricted projection. Thus, we get the following result, for every canonical relation $R \in \mathrm{Inv}(\mathrm{Pol}(\Gamma))$ we have $R \in \langle R \rangle_{re}$. ∎

Very recently, there has been an increasing interest in characterizing some "closed" sets of boolean functions [30, 73, 32, 42]. Pippenger [73] introduced Galois connection as a tool of study of these problems and he proved a characterization of some noteworthy classes of Boolean functions adapting the proof in Geiger [36]. This approach has been useful to characterize other classes of Boolean functions (see Hellerstein [42], for example).

Theorem 15 has interesting consequences. Let F be any class of operations, then $\mathrm{Inv}(F) = \mathrm{Inv}(\mathrm{Pol}(\mathrm{Inv}(F))) = \langle \mathrm{Inv}(F) \rangle_r$ is a co-clone. Furthermore, if F is a clone, then $\mathrm{Pol}(\mathrm{Inv}(F)) = \langle F \rangle_o = F$. Analogous results are obtained for classes of relations. Thus we have

Theorem 16 *(a) If F is any class of operations, then* $\mathrm{Pol}(\mathrm{Inv}(F)) = F$ *iff F is a clone. (b) If Γ is any class of relations, then* $\mathrm{Inv}(\mathrm{Pol}(\Gamma)) = \Gamma$ *iff Γ is a co-clone.*

Conversely, Theorem 15 can be deduced from Theorem 16. In fact, every theorem reflects a different perspective of the same mathematical concept. Theorem 15 focus in closure operators and Theorem 16 focus in closure systems.

5.6 Some Families of Operations

In this section we introduce some families of operations that will be of intensive use in next chapters. Furthermore, we prove useful properties.

5.6.1 Coset Generating Operations

We define a new family of operations.

Definition 11 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 [84] 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 17 *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)*** $\Longrightarrow$ ***(a)].*** A direct application of the properties of a group yields that

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

(ii)[(a) $\Longrightarrow$ ***(b)].*** Choose an element $1 \in A$ arbitrary, 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

$$\begin{aligned}(x^{-1})^{-1} &= \varphi(1, x^{-1}, 1) = \varphi(x \cdot x^{-1}, x^{-1}, 1) = \\ &= \varphi(\varphi(x, 1, x^{-1}), x^{-1}, 1) = \varphi(x, 1, 1) = x\end{aligned}$$

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 [84] for example). It is well known (see [49] 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 $|A|$.

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 18 *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.
$\boldsymbol{(i)[(a) \Longrightarrow (b)]}$ 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.
$\boldsymbol{(i)[(a) \Longrightarrow (b)]}$ Straightforward. ■

As a consequence of the previous result, for every nonempty relation $B \subseteq A^n$ preserved by an coset generating operation, $|B|$ (the number of tuples in B) divides $|A|^n$.

5.6.2 Near-unanimity Operations

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

Definition 12 [5] 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.

Definition 13 An n-ary relation R over A is said to be k-decomposable if it contains all n-tuples t such that $t_I \in R_I$ for all lists of indices I, from the set $\{1, 2, \ldots, n\}$ with $|I| \leq k$.

From [5] we have this useful property.

Theorem 19 *[5] Every relation R over A closed under a near-unanimity operation of arity k is $(k-1)$-decomposable.*

5.7 Boolean Case

Operations on a 2-element set, say $A = \{0, 1\}$ are boolean operations. The lattice of the clones (or equivalently co-clones) over the boolean domain was studied by Post, leading to a full description [76]. The proof is too long to be included here. We will give only the description of the lattice. Recall that a lattice is an algebra with two binary operations called *meet* (designated by $\wedge$) and *join* (defigned by $\vee$). In particular the lattice of clones is defined by $C_1 \wedge_o C_2 = C_1 \cap C_2$ and $C_1 \vee_o C_2 = \langle C_1 \cup C_2 \rangle_o$.

A usual way to describe a poset $(P, \leq)$ (and a lattice in particular) is by depicting a *diagram* with the relation *coverage*, where for every $a, b \in P$ we say that a covers b if $b \leq a$ and if c is an element in P such that $b \leq c \leq a$, then either $c = a$ or $c = b$. The diagram of the lattice of clones on $A = \{0, 1\}$, often called *Post's lattice* is depicted in Figure 5.1. The clones are labeled according their standard names.

To describe the vertices of the diagram, we have two possibilities. Every vertex can be described either as a clone, by an operational generating set, or as a co-clone, by a relational generating set. An element of a lattice a is *join irreducible* iff $a = b \vee c$ always implies $a = b$ or $a = c$. Dually, a is said to be *meet irreducible* iff $a = b \wedge c$ always implies $a = b$ or $a = c$. By virtue of the Galois connection between clones and co-clones described above, the lattice structure of clones and the lattice structure of co-clones is the same up to a "top-down" isomorphism defined by the mappings Inv and Pol. In consequence, the join operation $\vee_o$ in the clone lattice corresponds exactly to the meet operation $\wedge_r$ in the co-clone lattice and, conversely, the meet operation $\wedge_o$ in the clone lattice corresponds exactly to the join operation $\vee_r$ in the co-clone lattice. In Figure 5.1, the join irreducible clones of the diagram are denoted by $\smile$, while the meet irreducible clones are denoted by $\frown$.

Let F be any set of boolean operations. Since $\langle F\rangle_o = \bigvee_{f\in F}\langle f\rangle_o$, it follows that the join irreducible clones are generated by a single operation. Furthermore, it suffices to present a generating operation for each join irreducible clone of Post's lattice. Table 5.2 associates to every join irreducible clone $\mathcal{C}$, its generating operation $\varphi_{\mathcal{C}}$.

Similarly, for every set Γ of boolean relations, $\langle\Gamma\rangle_r = \bigvee_{R\in\Gamma}\langle R\rangle_r$. In consequence, the meet irreducible clones are determined by a single relation. Furthermore, it suffices to present a generating relation for each meet irreducible clone of Post's lattice. Table 5.3 associates to every meet irreducible clone $\mathcal{C}$, its generating relation $R_{\mathcal{C}}$.

For a n-ary boolean operation f define its *dual* dual(f) by

$$\mathrm{dual}(f)(x_1,\ldots,x_n) = \neg f(\neg x_1,\ldots,\neg x_n).$$

Obviously, $\mathrm{dual}(\mathrm{dual}(f)) = f$. Furthermore, f is self-dual iff $\mathrm{dual}(f) = f$. For a class F of boolean operations define $\mathrm{dual}(F) = \{\mathrm{dual}(f) : f \in F\}$. Notice that $\langle\mathrm{dual}(F)\rangle = \mathrm{dual}(\langle F\rangle)$.

Let F be a set of boolean operations. Then $\langle\mathrm{dual}(F)\rangle = \mathrm{dual}(\langle F\rangle)$. In consequence, the following clones are dual.

- O_5 and O_6,
- L_2 and L_3,
- M_2 and M_3,
- C_2 and C_3, and
- F_i^j and F_{i+4}^j, $j \in \{2, 3, \ldots, \infty\}, 1 \le i \le 4$.

The remaining clones are self-dual.

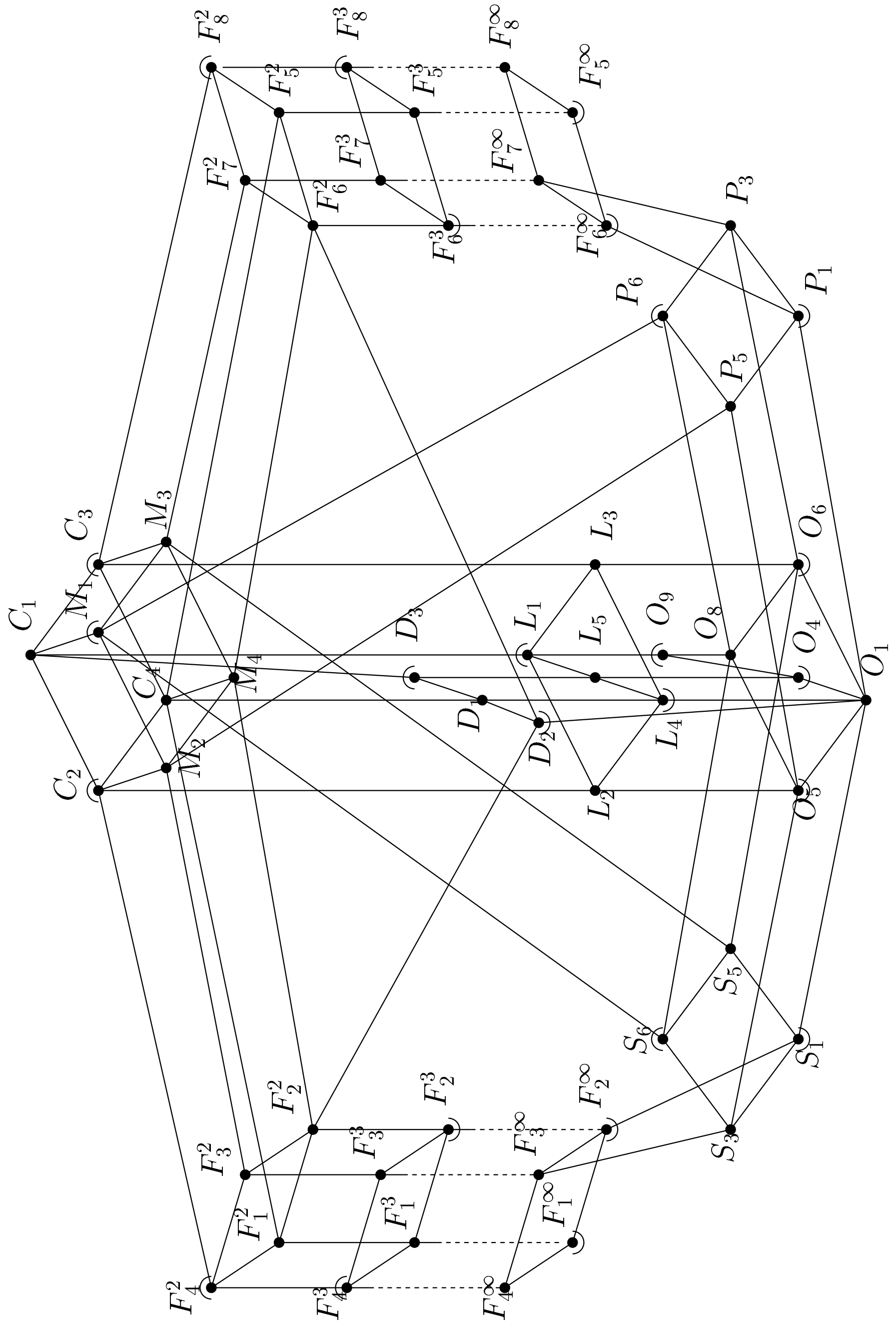

Figure 5.1: Post Lattice

D_2	$\varphi_{D_2}(x,y,z) = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$
F_1^∞	$\varphi_{F_1^\infty}(x,y,z) = x \vee (y \wedge \bar{z})$
F_2^∞	$\varphi_{F_2^\infty}(x,y,z) = x \vee (y \wedge z)$
F_2^i $(i \geq 2)$	$\varphi_{F_2^i}(x_1,\ldots,x_{i+1}) = \bigwedge_{j=1}^{i+1}(x_1 \vee \cdots \vee x_{j-1} \vee x_{j+1} \vee \cdots \vee x_{i+1})$
F_5^∞	$\varphi_{F_5^\infty}(x,y,z) = x \wedge (y \vee \bar{z})$
F_6^∞	$\varphi_{F_6^\infty}(x,y,z) = x \wedge (y \vee z)$
F_6^i $(i \geq 2)$	$\varphi_{F_6^i}(x_1,\ldots,x_{i+1}) = \bigvee_{j=1}^{i+1}(x_1 \wedge \cdots \wedge x_{j-1} \wedge x_{j+1} \wedge \cdots \wedge x_{i+1})$
L_4	$\varphi_{L_4}(x,y,z) = x \oplus y \oplus z$
O_1	$\emptyset$
O_4	$\varphi_{O_4}(x) = \bar{x}$
O_5	$\varphi_{O_5}(x) = 1$
O_6	$\varphi_{O_6}(x) = 0$
P_1	$\varphi_{P_1}(x,y) = x \wedge y$
S_1	$\varphi_{S_1}(x,y) = x \vee y$

Figure 5.2: Generating operations for meet irreducible clones

C_1	$\emptyset$
C_2	$R_{C_2} = \{\langle 1 \rangle\}$
C_3	$R_{C_3} = \{\langle 0 \rangle\}$
D_3	$R_{D_3} = \{\langle 0,1 \rangle, \langle 1,0 \rangle\}$
M_1	$R_{M_1} = \{\langle 0,0 \rangle, \langle 0,1 \rangle, \langle 1,1 \rangle\}$
L_1	$R_{L_1} = \{t \in \{0,1\}^4 : \lvert t \rvert_1 \mod 2 = 0\}$
O_9	$R_{O_9} = \{t \in \{0,1\}^8 : \lvert t \rvert_1 \mod 4 = 0\}$
F_4^i $(i \geq 2)$	$R_{F_4^i} = \{0,1\}^i - \langle 0,\ldots,0 \rangle$
F_8^i $(i \geq 2)$	$R_{F_8^i} = \{0,1\}^i - \langle 1,\ldots,1 \rangle$
P_6	$R_{P_6} = \{\langle 0,0,0 \rangle, \langle 0,0,1 \rangle, \langle 0,1,0 \rangle, \langle 1,0,0 \rangle, \langle 1,1,1 \rangle\}$
S_6	$R_{S_6} = \{\langle 0,0,0 \rangle, \langle 0,1,1 \rangle, \langle 1,0,1 \rangle, \langle 1,1,0 \rangle, \langle 1,1,1 \rangle\}$

Figure 5.3: Generating relations for meet irreducible co-clones

Chapter 6

Closure Operations and Width 1 Problems

6.1 Introduction

The constraint satisfaction problem provides a framework in which it is possible to express, in a natural way, many combinatorial problems encountered in artificial intelligence and elsewhere. A constraint satisfaction problem is represented by a set of variables, a domain of values for each variable, and a set of constraints between variables. Generally, a constraint C is represented by two components: a *scope* S that expresses the set of variables constrained and a *relation* R which expresses the combination of values allowed for those variables. The aim of a constraint satisfaction problem is then to find an assignment of values to the variables that satisfies the constraints.

Since solving a general constraint satisfaction problem is known to be NP-complete [13, 61], it arises naturally the important question of identifying restrictions to the general problem which are sufficient to ensure tractability. Such restrictions may either involve the scope in the constraints, in other words, which variables may be constrained with other variables [26, 35, 39, 65, 66], or they may involve the relation in the constraints, in other words, which combinations of values are allowed for values which are mutually constrained [15, 49, 52, 59, 65, 87, 88].

In this chapter, we focus in the second approach. More precisely we are interested in characterizing the complexity of solving constraint satisfaction problems in which every constraint belongs to a fixed set Γ called *basis*. We will denote this family of problems by CSP(Γ). We are interested precisely in this class of problems because it fits exactly in the framework introduced in Chapter 2. More precisely, the computational problem CSP(Γ) is identical to decide the satisfiability of generalized formulas over Γ.

Jeavons, Cohen and Gyssens in [49, 50] introduced a novel framework for studying this class of constraint satisfaction problems. The approach started in [49, 50] relies on the fact that the tractability of constraint satisfaction problems using certain relations depends on the set of polymorphisms of the basis Γ of the set of relations used to build the problem. This approach has lead to the identification of several tractable classes [49, 50, 51, 52, 48]. In this chapter we introduce the connection between constraint satisfaction problems and Clones and we give an overview of the tractable classes that have been identified using this approach.

Local consistency methods [61, 33, 34, 48, 28] have been intensively studied as a fundamental tool for solving constraint problems. Briefly, the underlying idea in local consistency methods is the following: when a constraint network is inconsistent, this is probably due to the existence of an inconsistent subnetwork of reduced size. It is sometimes possible to decide the existence of a solution of a constraint satisfaction problem by checking the consistency of all the subproblems up to a certain size. Since there exists problems which can not be solved by local consistency methods, there is a large body of work [35, 25, 28, 48] which derives conditions that guarantee that a problem can be solved by a certain local consistency method.

In this chapter we introduce a different family of local consistency methods called (j, k)-consistency, derived from the concept of bounded width Datalog programs [31] that is related with the standard notion of k-consistency as defined in [35]. In particular we will be interested in $(1, k)$-consistency, which can be regarded as the natural extension of arc-consistency [61] to non-binary problems. We refer to the class of problems solvable by enforcing $(1, k)$-consistency as width 1 problems. The class width 1 contains some previously known tractable families of problems including Horn, constant and ACI Problems [81, 51]. The main result of this chapter is a characterization of width 1 problems in terms of polymorphisms. That is, we present an algebraic condition (in the sense of [51]) on the relations used to build constraint problems which ensure that any problem built using these relations can be solved in polynomial time by enforcing $(1, k)$-consistency for some k. Furthermore, we show that this condition is also necessary.

With this new characterization we revisit some already known tractable families, as constant and ACI problems [51], and we see that they fit into this common scheme, in other words, we prove that they are particular case of width 1 problems. Furthermore, we derive a new tractable class called CSCC (from Constant Semiprojection Commutative Conservative) using purely algebraic arguments.

The motivation of these results is an alternative characterization of width 1 problems due to Feder and Vardi [31]. Feder and Vardi [31], using concepts from Datalog programs and group theory identify three main families of tractable problems, referred to as *subgroup*, *bounded strict width* and *width* 1 problems which include all previously known tractable problems. In [48], the family of strict width problems was characterized in terms of closure operations. An important subclass of subgroup problems called *affine* problems was introduced in [51]. In this chapter we extend the results in [51] in order to cover subgroup problems and we characterize the class of width 1 problems, completing the research work started in [48].

6.2 Constraint Satisfaction Problems

A constraint satisfaction problem is a natural way to express simultaneous requirements for values of variables. More precisely,

Definition 14 An instance of a *constraint satisfaction problem* consists of:

- a finite set of variables, V. For the sake of simplicity we assume $V = \{x_1, x_2, \ldots x_n\}$;
- a finite domain of values, D;
- a finite set of constraints $\{C_1, \ldots, C_q\}$; each constraint C_i $(1 \leq i \leq q)$ is a pair (s_i, R_i) where:
  - s_i is a tuple of variables of length k_i, called the *constraint scope*; and
  - R_i is an k_i-ary relation over D, called the *constraint relation.*

For each constraint (s_i, R_i), the tuples in R_i indicate the allowed combinations of simultaneous values for the variables in s_i. The length of s_i, and of the tuples in R_i, is called the *arity* of the constraint.

A *solution* to a constraint satisfaction problem instance is a function from the variables to the domain such that the image of each constraint scope is an element of the corresponding constraint relation.

Deciding whether or not a given problem instance has a solution is NP-complete in general, even when the constraints are restricted to binary constraints [61] or the domain of the problem has size 2 [13]. However, by imposing restrictions on the constraint interconnections (see [26, 35, 39, 65, 66]), or the form of the constraints (see [15, 49, 52, 59, 65, 87, 88]), it is possible to obtain restricted versions of the problem that are tractable. We address the problem of determining possible restrictions on the form of the constraints that ensure tractability. Thus, we restrict ourselves to the study of problems defined from some fixed set (or basis) of relations, and the associated complexity of the constraint problems using relations from that set.

Definition 15 For any set of relations Γ, $\mathrm{CSP}(\Gamma)$ is defined to be the class of decision problems with:

- Instance: A constraint satisfaction problem instance $\mathcal{P}$, in which all constraint relations are elements of Γ.
- Question: Does $\mathcal{P}$ have a solution?

This family of problems turns out to be equivalent to some instance of generalized formulas over a basis. More precisely, problem $\mathrm{CSP}(\Gamma)$ can be reformulated, using the definitions introduced in Section 2.1.1, as $\mathrm{SAT}(\mathrm{Form}(\Gamma))$. The equivalence between formulas in $\mathrm{Form}(\Gamma)$ and constraint networks in $\mathrm{CSP}(\Gamma)$ is immediate: every constraint $\langle (x_1, \ldots, x_k), R\rangle$ in a constraint network is associated to the occurrence of the term $R(x_1, \ldots, x_k)$ in the associated formula. For the sake of consistency with the existing CSP literature, along these pages, unless explicitly stated, we will follow the standard notation for constraint satisfaction problems, as introduced above.

6.3 Relationship with Universal Algebra

To our knowledge even though there exists a large collection of individual results about the complexity of $\mathrm{CSP}(\Gamma)$, there have only been two attempts to study the complexity of $\mathrm{CSP}(\Gamma)$ in a uniform way. The first attempt, due to Feder and Vardi [31], uses an approach based in Datalog Programs and

Group Theory and produced, among other important results, a classification of the known tractable classes in three families of basis, called *width* 1, *bounded strict width*, and *subgroup problems*. The second attempt was originally defined by Jeavons et al. [49, 50] and focuses on the algebraic properties of constraints and its link to Clone Theory. We follow the second approach. In the rest of this section we will introduce the link between constraint satisfaction problems over a fixed basis and Clone Theory proposed in [51, 47].

In this section, for simplicity, we will consider the formulation of $\mathrm{CSP}(\Gamma)$ in terms of generalized formulas ($\mathrm{SAT}(\mathrm{Form}(\Gamma))$). First, notice that existential quantification alone does not increase the computational complexity of the satisfiability problem since a existentially quantified formula has a solution iff the associated formula obtained eliminating the existential quantification has a solution. Thus, $\mathrm{SAT}(\mathrm{Form}(\Gamma))$ and $\mathrm{SAT}(\exists\text{-}\mathrm{Form}(\Gamma))$ are computationally equivalent. Recall that $\exists\text{-}\mathrm{Rel}(\Gamma)$ is the set of relations associated to formulas in $\exists\text{-}\mathrm{Form}(\Gamma)$. For clarity, let us interpret $\exists\text{-}\mathrm{Rel}(\Gamma)$ in terms of constraint satisfaction problems:

Associated to every instance of a constraint satisfaction problem $\mathcal{P}$ with set of variables V and domain D, and every list $I = (x_{i_1}, \ldots, x_{i_l})$ of variables in V we have a relation $R_{\mathcal{P},I}$ over D with arity l defined by

$$R_{\mathcal{P},I} = \{\langle h(x_{i_1}), \ldots, h(x_{i_l})\rangle : h \text{ is a solution of } \mathcal{P}\}$$

Relation $R_{\mathcal{P},I}$ characterizes the relation represented by constraint satisfaction problem $\mathcal{P}$ restricted to the variables in I.

Similarly, for every set of relations Γ over D, we can consider the set of relations represented by all the subinstances of problems in $\mathrm{CSP}(\Gamma)$ as a characterization of the expressivity power of the set of relation constraints Γ. Thus, the operator $\exists\text{-}\mathrm{Rel}(\Gamma)$ from sets of relations over D into itself can be defined alternatively by:

$$\exists\text{-}\mathrm{Rel}(\Gamma) = \{R_{\mathcal{P},I} : \mathcal{P} \in \mathrm{CSP}(\Gamma) \text{ with set of variables } V \text{ and } I \subseteq V\}.$$

In a certain sense, whereas $\mathrm{CSP}(\Gamma)$ represents the syntaxis of constraint satisfaction instances over Γ, the set $\exists\text{-}\mathrm{Rel}(\Gamma)$ represents its semantical counterpart.

The class $\exists\text{-}\mathrm{Rel}(\Gamma)$ turns out to be closely related to $\langle\Gamma\rangle_r$.

First, we will see that for every set of relations Γ over D containing the equality relation eq_D, the co-clone generated by Γ, $\langle\Gamma\rangle_r$ is contained in $\exists\text{-}\mathrm{Rel}(\Gamma)$. We have to check that closure conditions $\mathrm{CC}_1 - \mathrm{CC}_3$ are satisfied by $\exists\text{-}\mathrm{Rel}(\Gamma)$.

CC_1 Immediate. Relation D^2 is associated to the formula over two free variables x_1 and x_2 in which does not occur any relation in Γ. Relation eq_D is associated precisely to the formula $\Phi = \mathrm{eq}_D(x_1, x_2)$, since $\mathrm{eq}_D \in \Gamma$.

CC_2 For any $R, S \in \exists\text{-}\mathrm{Rel}(\Gamma)$ with arity n and m respectively, let Φ and Ψ formulas in $\exists\text{-}\mathrm{Form}(\Gamma)$ over $\{x_1, \ldots, x_n\}$ and $\{x_{n+1}, \ldots, x_{n+m}\}$ respectively with $[\Phi] = R$ and $[\Psi] = S$. Then, formula $(\Phi \wedge \Psi) \in \exists\text{-}\mathrm{Rel}(\Gamma)$ has associated relation $[\Phi \wedge \Psi] = R \times S$. Thus $R \times S \in \exists\text{-}\mathrm{Rel}(\Gamma)$.

CC_3 For any n-ary relation R in $\exists\text{-}\mathrm{Rel}(\Gamma)$ and any list of indices $I = (i_1, \ldots, i_m)$ over $\{1, \ldots, n\}$, let $\Phi \in \exists\text{-}\mathrm{Rel}(\Gamma)$ be a formula over the free variables $\{x_1, \ldots, x_n\}$ with $[\Phi] = R$. Clearly, there exists some formula $\Psi \in \exists\text{-}\mathrm{Form}(\Gamma)$ obtained taking Φ, renaming some variables and quantifying existentially some variables, such that $[\Psi] = R_I$.

Thus, $\langle\Gamma\rangle_r \subseteq \exists\text{-}\mathrm{Rel}(\Gamma)$. Conversely, the relation R associated to every formula Φ in $\exists\text{-}\mathrm{Form}(\Gamma)$ is exactly the projection (existential quantification) of the conjunction of the relations occurring in Φ. In consequence R can be obtained from some relations in Γ by a sequence of cylindrifications in order to add the needed inessential variables, followed by a sequence of rotations and transpositions in order to constraint the right variables and finally a projection to eliminate variables existentially quantified. Thus, $\exists\text{-}\mathrm{Rel}(\Gamma) \subseteq \langle\Gamma\rangle_r$. So, we have just proved the following result

Theorem 20 *For every set of relations Γ over D, $\exists\text{-}\mathrm{Rel}(\Gamma \cup \{\mathrm{eq}_D\}) = \langle\Gamma\rangle_r$*

In fact, equality relation eq_D is not relevant when we consider the satisfiability problem.

Theorem 21 *For any Γ over finite domain D, $\mathrm{SAT}(\exists\text{-}\mathrm{Form}(\Gamma \cup \{\mathrm{eq}_D\}))$ is polynomial-time reducible to $\mathrm{SAT}(\exists\text{-}\mathrm{Form}(\Gamma))$.*

Proof. Let Φ be any formula in $\exists\text{-}\mathrm{Form}(\Gamma \cup \{\mathrm{eq}_D\})$. Modify Φ to obtain a formula Ψ in $\exists\text{-}\mathrm{Form}(\Gamma)$ as follows: For every occurrence of the identity relation $\mathrm{eq}_D(x_1, x_2)$ in Φ replace all the occurrences of x_2 in all the relations R occurring in Φ by x_1. Then remove variable x_2 and the relation $\mathrm{eq}_D(x_1, x_2)$ in Φ. Clearly, this construction can be carried out in polynomial time in the size of Φ, and Ψ has a solution if and only if Φ has a solution. Hence, we have established a polynomial-time reduction from $\mathrm{SAT}(\exists\text{-}\mathrm{Form}(\Gamma \cup \{\mathrm{eq}_D\}))$ to $\mathrm{SAT}(\exists\text{-}\mathrm{Form}(\Gamma))$. ■

As a consequence of the previous results we have that the complexity of $\mathrm{SAT}(\exists\text{-}\mathrm{Form}(\Gamma))$, for a finite Γ, is characterized by the co-clone generated by Γ, $\langle\Gamma\rangle_r$ or equivalently by its clone of polymorphisms $\mathrm{Pol}(\Gamma)$.

Theorem 22 *Let Γ and Γ_0 be sets of relations over D such that Γ_0 is finite and $\mathrm{Pol}(\Gamma) \subseteq \mathrm{Pol}(\Gamma_0)$. Then $\mathrm{CSP}(\Gamma_0)$ is polynomially reducible to $\mathrm{CSP}(\Gamma)$.*

Proof. We show a polynomial-time reduction from $\mathrm{Pol}(\Gamma_0)$ to $\mathrm{Pol}(\Gamma)$. The reduction has five steps

$$\begin{aligned}\mathrm{CSP}(\Gamma_0) &\leq_m \mathrm{SAT}(\exists\text{-}\mathrm{Form}(\Gamma_0)) \leq_m \mathrm{SAT}(\exists\text{-}\mathrm{Form}(\Gamma_0 \cup \{\mathrm{eq}_D\})) \leq_m \\ &\leq_m \mathrm{SAT}(\exists\text{-}\mathrm{Form}(\Gamma \cup \{\mathrm{eq}_D\})) \leq_m \mathrm{SAT}(\exists\text{-}\mathrm{Form}(\Gamma)) \leq_m \mathrm{CSP}(\Gamma)\end{aligned}$$

The first, second, and fifth reductions are trivial. The third reduction follows from the fact that if $\mathrm{Pol}(\Gamma) \subseteq \mathrm{Pol}(\Gamma_0)$ then $\langle\Gamma_0\rangle_r \subseteq \langle\Gamma\rangle_r$. Then by virtue of Theorem 20, $\exists\text{-}\mathrm{Rel}(\Gamma_0 \cup \{\mathrm{eq}_D\}) \subseteq \exists\text{-}\mathrm{Rel}(\Gamma \cup \{\mathrm{eq}_D\})$. Then, given any formula Φ in $\exists\text{-}\mathrm{Form}(\Gamma_0 \cup \{\mathrm{eq}_D\})$ it is possible to replace every occurrence of a relation R in Φ by a formula Ψ_R in $\exists\text{-}\mathrm{Form}(\Gamma \cup \{\mathrm{eq}_D\})$ with the same satisfying assignments obtaining an equivalent formula in $\exists\text{-}\mathrm{Form}(\Gamma \cup \{\mathrm{eq}_D\})$. 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 $|\Phi|$. The fourth reduction is basically the content of Theorem 21. ■

So, the computational complexity of $\mathrm{CSP}(\Gamma)$ is characterized by its clone of polymorphisms, i.e., $\mathrm{Pol}(\Gamma)$. Then, the case analysis is reduced since, instead of consider all possible finite sets of relations, we only have to consider the relational co-clones finitely generated.

Furthermore, it is justified to talk about the complexity of a clone of operations.

Definition 16 Let D be a finite set and let F be a clone of operations over D. We say that F is *tractable* if for every finite set Γ of relations over D with $F \subseteq \mathrm{Pol}(\Gamma)$, $\mathrm{CSP}(\Gamma)$ is solvable in polynomial time. Let F be a set of operations over D. We say that F is tractable if $\langle F\rangle_o$ is tractable. Let $\mathbf{D} = (D; F)$ be a finite algebra. We say that $\mathbf{D}$ is tractable if the clone of its term operations $T(\mathbf{D})$ is tractable.

A remark: We have three related concepts with respect the complexity of solving constraint satisfaction problems: sets of finite relations or basis, clones of operations and algebras. Probably the most used notion will be that of a clone, but for convenience, we will move freely from one to another.

As a direct consequence of Theorem 22 we have:

Corollary 5 *Let $F \subseteq F_0$ clones on D. If F is tractable then F_0 is tractable.*

6.3.1 Previous Work

The relationship between constraint satisfaction problems and clones of operations shown in the previous section has lead to the following research project: Let D be an arbitrary domain. Identify all the tractable clones on D.

Actually, the problem can be reduced considerably, since by virtue of Theorem 22 we know that if a clone C is tractable then any other clone C' with $C \subseteq C'$ is also tractable. Technically it is said that the tractable clones is an order filter in the lattice of clones (See [64] for more details about lattices). Thus we are interested in identifying the minimal clones in the order filter of tractable clones. Before the work in this thesis, only four (not all necessarily minimal) families of tractable clones have been identified. Actually all of them are generated by a single operation: (**1**) near-unanimity operations, (**2**) coset generating operations, (**3**) ACI operations and (**4**) constant operations.

Recall that a clone F is tractable if for every finite set of relations Γ such that $F \subseteq \mathrm{Pol}(\Gamma)$ is tractable. In consequence, when we are talking about families of tractable clones (or operations), really we are talking about the families of finite basis associated to them. Let us define them: We say say that a basis Γ is near-unanimity (resp. coset generating, ACI, constant) if $\mathrm{Pol}(\Gamma)$ contains a near-unanimity (resp. coset generating, ACI, constant operation).

Notice that every one of this families is defined by the closure under some operation of bounded arity. In consequence we have the following interesting property:

Theorem 23 *The problem of deciding whether a finite set of relation Γ is is in any of the previous classes is decidable.*

Near-Unanimity Operations

Using the decomposability property of the relations closed under a near-unanimity operation (Theorem 19) it is easy to derive an efficient algorithm for CSP(Γ) when Γ is preserved by a near-unanimity operation. In [48] it is shown that a basis Γ is preserved by a near-operation of arity k iff it is possible of find a solution to any constraint instance $\mathcal{P}$ in CSP(Γ) by a backtrack-free search after enforcing $(k-1)$-consistency. The same result was previously obtained in [31] using a different approach based in Datalog Programs. The basis preserved by a near-unanimity operation in the boolean domain are precisely the bijunctive, weakly-monotone and weakly-antimonotone basis.

Coset Generating Operations

The class of coset generating operations was identified by Feder and Vardi [31] without making an explicit connection to preserving operations. The class of problems identified in [31], called *subgroup problems*, consists in the problems $\mathcal{P}$ obtained in the following way: fix a group operation $(D; \cdot, ^{-1}, 1)$ on the domain D, then every n-ary ($n \geq 0$) constraint relation in $\mathcal{P}$ is a coset of some subgroup of the group power $(D; \cdot, ^{-1}, 1)^n$. It is clear from the definition and Theorem 18, that subgroup problems correspond to problems over a basis preserved by a coset generating operation.

The class of subgroup problems was partially characterized in terms of preserving operations in [51]. More precisely, the class identified in [51] corresponds to the subclass of subgroups problems over a boolean group. The basis preserved by a coset generating operation in the boolean domain are the affine basis in Schaefer's classification.

We mention here that Feder and Vardi [31] proposed some ways to extend subgroup problems that are not covered by closure under a coset generating operation.

ACI operations

A binary operation $f : D^2 \rightarrow D$ is an *ACI operation* if f is associative, commutative and idempotent. Tractability of ACI operations is shown in [51]. The basis preserved by an ACI operation in the boolean domain are the horn and antihorn basis.

Constant operations

A constant operation $f : D \rightarrow D$ is an unary operation such that $f(x) = d$ for all $x \in D$ for a fixed $d \in D$. In consequence, every nonempty relation

R preserved by f contains tuple $(d, \ldots, d)$. Tractability of constant operations follows from this simple property. The basis preserved by a constant operation in the boolean domain are the 0-valid and 1-valid basis.

6.4 Local Consistency

Local consistency methods are refutation methods, similar in aim to the resolution method for propositional formulas in conjunctive normal form. Different notions of local consistency have been defined [61, 33, 34, 48, 28] but all of them fit into this common scheme: A local consistency method takes an instance $\mathcal{P}$ and adds all the constraints that appear implicitly in $\mathcal{P}$ (therefore, eliminating no solution) up to a certain level. A constraint $\langle s, \emptyset \rangle$ with empty constraint relation is called *empty constraint*. If during the process of enforcing consistency, some empty constraint is added, then instance $\mathcal{P}$ has obviously no solution. Unfortunately, the reciprocal is not always true, i.e., the absence of the empty constraint does not imply in general the existence of a solution. So, it is an interesting topic of research to establish under which conditions over the set of constraint relations Γ, establishing a certain level of local consistency guarantees the existence of a solution.

Definition 17 Let $\mathcal{P}$ be a constraint satisfaction instance with set of variables V, domain D, and constraint set C. For any subset W of V, the *restriction of* $\mathcal{P}$ *to* W, denoted $\mathcal{P}^*_W$ is the problem instance with set of variables W and domain D, where the constraints are obtained from the constraints of $\mathcal{P}$ eliminating all the constraints with constraint scope not completely contained in W. That is, the constraint (s, R) is in $\mathcal{P}^*_W$ if and only if (s, R) is in $\mathcal{P}$ and every element of the tuple s is in W.

Definition 18 A constraint satisfaction problem instance $\mathcal{P}$ with set of variables V is said to be (j, k)*-consistent* $(0 \leq j \leq k)$ if for any sets of variables W, W', such that $W \subseteq W' \subseteq V$, containing at most j and k variables respectively, any solution to $\mathcal{P}^*_W$ can be extended to a solution to $\mathcal{P}^*_{W \cup W'}$.

Informally, a problem is (j, k)-consistent if every partial solution on any set of at most j variables can be extended to a partial solution on any superset containing at most k variables.

Since the constraint literature contains few similar notions of consistency we believe that it is appropriate to enclose here a brief discussion of the more popular notions of consistency. Informally we can say that the differents notions of consistency are characterized by the way in which they define a subproblem. Roughly, consistency notions can be divided in two main types: (1) *variable-based* consistency and (2) *relation-based* consistency.

In variable-based consistency a subproblem is specified as the subproblem containing all the restrictions "concerning" a set of variables. Variable-based consistency has received more attention (see [61, 33, 34] for example) than its relational counterpart. We have different consistency notions depending on how "concerning" is formalized. Very often, the subproblem "concerning" a set of variables W is defined as the subproblem containing all the constraints with scope strictly contained in W, as in the definition above. This approach is due to [61] and was generalized to sets of variables with arbitrary size in [33]. This notion of consistency in closely related to the notion defined here. More precisely, k-consistency in the sense of [33] coincides exactly with $(k-1,k)$-consistency, as defined above. In some other cases, the problem "concerning" a set of variables W has been defined as the problem containing the projection of all the constraints over W (see [48], for example). In fact, both definitions are equivalent for the purposes of this chapter. We choosed this particular notion of consistency to simplify the proofs. Finally, in relation-based consistency (see [28] for example), a subproblem is specified by a set of restrictions rather than variables.

Given a CSP instance $\mathcal{P}$, there exists some instance $\mathcal{P}'$ such that (1) Every constraint in $\mathcal{P}$ is in $\mathcal{P}'$, (2) $\mathcal{P}'$ is (j,k)-consistent, and (3) instances $\mathcal{P}$ and $\mathcal{P}'$ have the same set of solutions. Instance $\mathcal{P}'$ is referred to as a (j,k)-consistent instance associated to $\mathcal{P}$. We say that $\mathcal{P}'$ is obtained from $\mathcal{P}$ by enforcing (j,k)-consistency. The CSP literature [65, 61, 33, 14] contains some efficient methods to enforce certain level of consistency. Figure 6.4 shows a simple brute-force algorithm, called $\mathrm{Cons}_{(j,k)}$ (j,k fixed), that enforces (j,k)-consistency.

Basically, the algorithm looks for subsets $W \subseteq W'$ of variables falsifying the (j,k)-consistency condition (Definition 18) and adds the correspondent constraint until the instance $\mathcal{P}'$ satisfies the (j,k)-consistency condition and no constraint has to be added. For fixed values of j,k, procedure $\mathrm{Cons}_{(j,k)}$ runs in time polynomial to the size of the input instance $\mathcal{P}$.

Notice that the constraints added by algorithm $\mathrm{Cons}_{(j,k)}$ are minimal in the sense that for every (j,k)-consistent instance $\mathcal{P}'$ associated to $\mathcal{P}$ and every variable x_i $(1 \leq i \leq n)$, every solution of ${\mathcal{P}'}^*_{\{x_i\}}$ is a solution of $(\mathrm{Cons}_{(j,k)}(\mathcal{P}))^*_{\{x_i\}}$.

Procedure $\mathrm{Cons}_{(j,k)}$
Input: $\mathcal{P}$
$\mathcal{P}' = \mathcal{P}$
repeat
    for every subset W with $|W| \leq j$
        for every superset W' of W with $|W'| \leq k$
            for every tuple $s = \langle x_1, \ldots, x_i \rangle$ with variables in W
                let R be the projection of ${\mathcal{P}'}^*_{W'}$ over s, i.e.,
                    $R = \{\langle \mu(x_1), \ldots, \mu(x_i) \rangle : \mu \text{ solution of } {\mathcal{P}'}^*_{W'}\}$
                (* R contains all the assignment over s that can be extended to a solution of ${\mathcal{P}'}^*_{W'}$ *)
                if $\langle s, R \rangle$ not in $\mathcal{P}'$ then add $\langle s, R \rangle$ to $\mathcal{P}'$.
until no constraint has been added.
return $\mathcal{P}'$.

Figure 6.1: Brute-force algorithm enforcing j, k-consistency

Furthermore, for every constraint $\langle s, R \rangle$ added by the algorithm, its associated relation R is the projection of subproblem ${\mathcal{P}'}^*_{W'}$ over s. Since every relation associated to a problem (or equivalently subproblem) with constraint relations in Γ can be obtained by a sequence of of cartesian products and equality selections using relations in Γ. We have that if $\mathcal{P} \in \mathrm{CSP}(\Gamma)$, then $R \in \langle \Gamma \rangle_r$.

Definition 19 A CSP instance $\mathcal{P}$ is said to have *width* (j, k) if $\mathcal{P}$ has a solution if and only if every (j, k)-consistent instance associated to $\mathcal{P}$ does not contain the empty constraint relation $\langle s, \emptyset \rangle$. Similarly, a set of relations Γ over D is said to have width (j, k) if every CSP instance $\mathcal{P}$ in $\mathrm{CSP}(\Gamma)$ has width (j, k). Furthermore Γ is said to have width j if it has width (j, k) for some k.

Hence, a set of relations Γ has width (j, k) if and only if every problem in $\mathrm{CSP}(\Gamma)$ is solvable in polynomial time by enforcing (j, k)-consistency (using the previous algorithm for example).

Feder and Vardi [31] gave an alternative characterization of these concepts in terms of Datalog Programs. Datalog Programs are far beyond the scope of this chapter. Nevertheless, we will only note here, that the notion of width defined above is exactly equivalent to the notion of width, as defined in [31].

6.5 Width 1 Problems

In this section we introduce and prove the main result of the chapter. We will see that the family of basis of width 1 is absolutely characterized in terms of closure operations. In other words, we will see that for every basis Γ, CSP(Γ) is solvable by enforcing $(1,k)$-consistency for some $k \geq 1$ if and only if some condition over the set of closure operations is satisfied. First we introduce the concept of set function.

A *set function* is any function $f : \mathcal{P}(D)/\emptyset \longrightarrow D$ where D is some set and $\mathcal{P}(D)$ is the set of subsets of D.

Associated to every set function f we have a *family of operations associated to f* $\{f_i : i = 1, \ldots, n\}$, where for every i, $f_i : D^i \longrightarrow D$ is given by

$$f_i(x_1, \ldots, x_i) = f(\{x_1, \ldots, x_i\}),$$

Let f be a set function over D and let R be any relation over D, we say that f *preserves* R, (or R is closed under f) if the family of operations associated to f preserves R (Definition 7).

Theorem 24 *Let Γ be a finite set of relations over D. Then the following conditions are equivalent:*

a.- The set Γ is width 1.

b.- Γ is closed under some set function.

The proof of the theorem uses the following result. Given any set of relations Γ, consider the constraint satisfaction problem $\mathcal{C}(\Gamma)$ defined as follows:

The variables of $\mathcal{C}(\Gamma)$ are the nonempty subsets A of D. For a relation R of arity k, impose $R(A_1, A_2 \ldots, A_k)$, the A_i not necessarily distinct, if for every $1 \leq i \leq k$ and every a_i in A_i there exist elements a_j in the remaining A_j such that $(a_1, a_2, \ldots, a_k) \in R$.

Theorem 25 *[31] If a set of relations Γ is width 1 then $\mathcal{C}(\Gamma)$ has a solution.*

Since, set functions establish a new family of tractable problems we should give them a name: A clone F is called a *set clone* if F containts the family of functions associated to some set function.

Proof of Theorem 24

(i)[(a) $\Longrightarrow$ (b)]. Let h be any solution of $\mathcal{C}(\Gamma)$ (the existence of such solution is guaranteed by Theorem 25) and let f_h be the set function defined as:

$$f^h(A_i) = h(A_i).$$

We will prove that f^h preserves Γ: Let m be any integer $m \geq 1$, let R be any k-ary relation in Γ, and let $t_1, t_2, \ldots, t_m$ (not necessarily different) tuples in R. Let A_i $(1 \leq i \leq k)$ be subsets of D given by:

$$A_i = \{t_l[i] : 1 \leq l \leq m\}, \quad 1 \leq i \leq k$$

By, construction, $R(A_1, A_2, \ldots, A_k)$ appears in $\mathcal{C}(\Gamma)$ and therefore h satisfies it. Then we have

$$\langle h(A_1), h(A_2), \ldots, h(A_k)\rangle = \langle f^h(A_1), f^h(A_2), \ldots, f^h(A_k)\rangle = f^h_m(t_1, \ldots, t_m),$$

which belongs to R.

(ii)[(b) $\Longrightarrow$ (a)]. Let Γ be a set of relations over D closed under a set function f. Let k be the maximum arity of the relations in Γ. We will prove that enforcing $(1, k)$-consistency is enough to decide satisfiability. Let $\mathcal{P}$ be any problem in CSP(Γ) and let $\mathcal{P}'$ be a problem obtained by enforcing $(1, k)$-consistency to $\mathcal{P}$. For every variable x, let D_x be the set of values that can be assigned to the variable x, i.e., the set of solutions of the problem $\mathcal{P}'$ restricted to x.

Consider the vector t, assigning to every variable x the value of the set function over D_x, that is,

$$t(x) = f(D_x)$$

We will see that t is a solution. Let $\langle (x_1, x_2 \ldots, x_m), R\rangle$ be any constraint in $\mathcal{P}$. Clearly, $D_{x_i} \subseteq \mathrm{pr}_i R$ $\;(1 \leq i \leq m)$. Consider now, the subset R' of R, obtained enforcing for every variable x appearing in the scope of the constraint to have a value in D_x. More formally,

$$R' = R \cap (D_{x_1} \times \cdots \times D_{x_m}).$$

Since, D_{x_i} $(1 \leq i \leq m)$ is obtained by enforcing $(1,k)$-consistency, we have $D_{x_i} = \mathrm{pr}_i\, R'$ $(1 \leq i \leq m)$. Let $t_1, t_2, \ldots, t_l$ be the tuples in R'. Since R is closed under f_l, we have:

$$f_l(t_1, t_2, \ldots, t_l) = \langle f(D_{x_1}), f(D_{x_2}), \ldots, f(D_{x_m})\rangle \in R.$$

So t satisfies R.

■

The characterization of width 1 problems in terms of closure operations is interesting theoretically but it is not absolutely satisfactory. First, if we want to use this characterization to check whether a given basis Γ is width 1 then we have to test the closure condition for an infinite family of operations.

Actually it is possible, for a given Γ, fix a bound in the arity of the operations that we have to consider, proving the decidability of the problem (the same result was established in [31] using different methods).

Theorem 26 *Let Γ be a finite set of relations over D, let m be the maximum number of tuples of any relation in Γ, and let f be a set function over D. If f_m preserves Γ then f preserves Γ.*

Proof.

Most of the next proofs have a similar structure. We will be interested in proving that if some operation h_1 (or set of operations) preserves a relation R then another operation h_2 preserves also R. We will prove that by showing that operation h_2 can be obtained by a sequence of compositions using h_1 and projection operations only. Then, since the set of closure operations of a relation contains projection operations and is closed under composition, operation h_2 preserves any relation preserved by h_1. We will say that h_2 is contained in any clone containing h_1.

In the present proof, we will see that every clone containing f_m, contains f_k $(k > 0)$, or in other words, f_k can be obtained by a composition using f_m and projections.

It is immediate to prove that, in general, for every set function f and every $k > 0$, f_k belongs to any clone containing f_{k+1}. Just consider the identity.

$$f_k(x_1, x_2, \ldots, x_k) = f_{k+1}(x_1, x_2, \ldots, x_k, x_k)$$

For the proof in the opposite direction, let R be any relation in Γ, let k be any integer greater than m, and let $t_1, t_2, \ldots, t_k$ tuples in R. Since $k > m$, some tuples are repeated. Consider a sublist with m tuples $t_{i_1}, t_{i_2}, \ldots, t_{i_m}$ in which we have deleted only repeated tuples, i.e., such that $\{t_j : 1 \leq j \leq k\} = \{t_{i_j} : 1 \leq j \leq m\}$. By the structure of the set functions we have:

$$f_k(t_1, t_2, \ldots, t_k) = f_m(t_{i_1}, t_{i_2}, \ldots, t_{i_m}) \in R$$

■

It would be interesting to get rid of this dependence on the number of tuples, that is, to prove that it is possible to verify whether or not a basis is closed under a set function by checking only the closure property up to a fixed point.

The previous assertion is true for the boolean case, since it is known that every clone in the boolean domain is finitely generated [76, 72] but it is false for higher domains. We present a counterexample.

Example 3 Let $D = \{0, 1, 2, \ldots, d\}$ be a domain with size $|D| > 2$. Consider the set function $f : \mathcal{P}(D)/\emptyset \longrightarrow D$ defined by:

$$f(S) = \begin{cases} 0 & \text{if } S = \{0,1\} \\ 2 & \text{otherwise} \end{cases}$$

For every m, consider the m-ary relation R_m containing all the tuples such that either (1) exactly one of its components is 1 and the remaining $m-1$ are 0, or (2) at least one of its components is 2. Clearly, f_k preserves R_m if and only if $k < m$. As a consequence, for any value of k there exists some basis given by $\Gamma_k = \{R_{k+1}\}$ such that f_k preserves Γ_k but f does not.

6.6 Applications of the Closure Conditions

Using the characterization of width 1 problems established in Theorem 24 it is possible reformulate some already known tractable classes and derive new ones.

6.6.1 Constant Operations

Constant operations are a particular case of width 1 problems. Since clones are closed under cylindrification, ie., addition of inessential variables, we have that for all $i > 0$, operation $f_i : D^i \longrightarrow D$ given by $f_i(x_1, \ldots, x_i) = f_1(x_1) = d$ belongs to any clone containing f_1. In consequence, any set of relations Γ closed under f_1 is closed under the set function $f : \mathcal{P}(D)/\emptyset \longrightarrow D$ given by $f(S) = d$.

6.6.2 ACI operations

Let f be an ACI operation. It is not difficult to see that, as a consequence of the properties of ACI operations, the set function g, given by

$$g(\{x_1, x_2, \ldots, x_i\}) = f(x_1, f(x_2, \ldots f(x_{i-1}, x_i) \ldots))$$

is well defined. For every i, g_i can be built by a sequence of compositions using f. Thus, any basis closed under f is also closed under g.

6.6.3 Class CSCC (Constant Semiprojection and Commutative Conservative)

In this section, we identify a new tractable family of set functions. First we need the following definitions:

Let D be a finite set. An m-ary operation f over D is conservative if for all $d_1, \ldots, d_m \in D$, $f(d_1, \ldots, d_m) \in \{d_1, \ldots, d_m\}$. A binary operation $f : D^2 \to D$ is a *CC operation* if f is commutative and conservative. A *semiprojection* g is an operation of rank $k \geq 3$ such that for some index $1 \leq i \leq k$, $g(x_1, \ldots, x_k) = x_i$ whenever $|\{x_1, \ldots, x_k\}| < k$, but g is not a projection. Furthermore, g is called *constant semiprojection* if $g(x_1, \ldots, x_k) = d$ for some fixed $d \in D$ when the semiprojection condition is not satisfied, i.e., $|\{x_1, \ldots, x_k\}| = k$.

Theorem 27 *Let Γ be a finite set of relations closed under some CC operation f and under some constant semiprojection g of arity 3. Thus* CSP(Γ) *is tractable.*

Proof. The proof has to main parts. Let g_k be a constant semiprojection of arity k where the variable projected is the first one and the constant is d. Then, for every $j > k$, the j-ary operation g_j given by:

$$g_j(x_1, \ldots, x_j) = \begin{cases} x_1 & \text{if } |\{x_1, \ldots, x_j\}| < k \\ d & \text{otherwise} \end{cases}$$

belongs to any clone containing g_k. That is, we will see that for every $j > k$, g_j can be obtained as a sequence of compositions using only g_k and projections.

We do it by induction. Assume that g_j with $j \geq k$ belongs to the clone. Operation g_{j+1} can be constructed from g_j in the following way:

$$g_{j+1}(x_1, \ldots, x_{j+1}) = g_j \ldots g_j(g_j(x_1, x_2, \ldots, x_j), x_2, \ldots \\ \ldots, x_{j-1}, x_{j+1}), \ldots), x_3, \ldots, x_{j+1})$$

Now consider the set function h given by:

$$h(S) = \begin{cases} f(S) & \text{if } |S| \leq 2 \\ d & \text{otherwise} \end{cases}$$

For every $k \geq 3$, operation h_k can be constructed by a sequence of compositions using f and g_k in the following way.

$$\begin{aligned} h_k(t_1, t_2 \ldots, t_k) &= f(\ldots f(f(g_k(t_1, t_2, \ldots, t_k), g_k(t_2, t_3, \ldots, t_1)), \ldots \\ &\quad g_k(t_{k-1}, t_k, \ldots, t_{k-2}), g_k(t_k, t_1, \ldots, t_{k-1})) \end{aligned}$$

In consequence, every clone containing f and g contains h_k for all $k \geq 3$. For $k \leq 2$ we have the easy equivalences: $h_2(x_1, x_2) = f(x_1, x_2)$ and $h_1(x_1) = f(x_1, x_1)$. Then set function h preserves any relations closed under f and g. ■

The class of constant operations and ACI operations have been previously shown to be tractable by other means. The class CSCC has not been previously identified. It is important to know if the tractability of the class CSCC is simply a consequence of a previously known class of tractable relations, or if a new class of tractable problems has been found. We show that the class CSCC does include problems which can not be accounted for in the framework of closure operations as in [51] by producing a relation which gives rise to tractable constraint problems but does not belong to any of the previously identified classes. As we have mentioned above, tractable classes of constraint satisfaction problems can be classified in four main families: **(1)** coset generating operations, **(2)** near-unanimity operations, and some set functions (as introduced in these pages). More precisely, the set functions already known are **(3)** constant operations and **(4)** ACI operations.

Thus, it is necessary to prove that there exists some CC operation φ and some constant semiprojection ϕ of arity 3 and some relation $\mathcal{R}$, all of them over the same domain D, such that φ and ϕ preserve $\mathcal{R}$, but none of the other known tractable classes preserves $\mathcal{R}$.

Let $D = \{0, 1, 2\}$ be the domain. Let $\varphi : D^2 \to D$ the CC operation with the Cayley table

	0	1	2
0	0	0	2
1	0	1	1
2	2	1	2

Operation φ is a tournament; it is known as *the scissors-paper-stone algebra.* Let $\mathcal{R}$ be the 3-ary relation with tuples

$$\begin{array}{ll} \{ & \langle 0,0,1\rangle \\ & \langle 0,2,1\rangle \\ & \langle 0,2,2\rangle \\ & \langle 1,0,2\rangle \quad \} \end{array}$$

It is immediate to verify that $\mathcal{R}$ is preserved by φ. Furthermore, since every column contains only two different values, $\mathcal{R}$ is preserved by all the semiprojections (and, in particular, relation $\mathcal{R}$ is preserved by all the constant semiprojections of arity 3).

The hard work is to prove that relation $\mathcal{R}$ is not closed under any other known tractable operation. We do a case analysis.

Coset Generating Operations

Recall that every n-ary relation R closed under a coset generating operation is a right coset of a group $(D; \cdot, ^{-1}, 1)^n$, and that, in consequence, the cardinality of R should divide $|D|^n$. It is clear from the previous observation that $\mathcal{R}$ is not preserved by a coset generating operation.

Near-Unanimity operations

Recall that an operation $f : D^k \to D$ $(k \geq 3)$, is a near-unanimity operation if for all $x, y \in D$,

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

To prove that none of the near-unanimity operations preserves $\mathcal{R}$ requires some detailed study. We study every arity k separately.

For $k = 3$ the analysis is simple. For every 3-ary near-unanimity operation m (also called majority operation) we have

$$m(\langle 0,0,1\rangle, \langle 0,2,2\rangle, \langle 1,0,2\rangle) = \langle 0,0,2\rangle \notin \mathcal{R}.$$

Thus, operation m does not preserve $\mathcal{R}$.

Now, assume $k \geq 4$. Let m be an k-ary near-unanimity operation over the domain D preserving $\mathcal{R}$. Since $m(0,2,\ldots,2) = 2$ and $m(0,0,\ldots,0,2) = 0$ there exists some integer $1 \leq n \leq k-2$ such that one of the following conditions is satisfied:

- $m(\overbrace{0,\ldots,0}^{n}, 2\ldots,2) = 1$, or
- $m(\overbrace{0,\ldots,0}^{n}, 2,\ldots,2) = 2$ and $m(\overbrace{0,\ldots,0}^{n+1}, 2,\ldots,2) = 0$

In the first case, we get a contradiction with

$$m(\overbrace{\langle 1,0,2\rangle,\ldots,\langle 1,0,2\rangle}^{n}, \langle 0,2,2\rangle,\ldots,\langle 0,2,2\rangle) = \langle x,1,2\rangle \in \mathcal{R}$$

(Impossible for any value for x).

In the second case we have,

$$m(\overbrace{\langle 1,0,2\rangle,\ldots,\langle 1,0,2\rangle}^{n}, \langle 0,0,1\rangle, \langle 0,2,2\rangle,\ldots,\langle 0,2,2\rangle) = \langle x,0,2\rangle \in \mathcal{R}$$

$$m(\overbrace{\langle 1,0,2\rangle,\ldots,\langle 1,0,2\rangle}^{n}, \langle 0,2,2\rangle, \langle 0,2,2\rangle,\ldots,\langle 0,2,2\rangle) = \langle x,2,2\rangle \in \mathcal{R}$$

Then, we get a contradiction since any value for x can satisfy both conditions.

Constant Operations

Immediate from the fact that $\mathcal{R}$ does not contain any tuple of the form (d,d,d).

ACI Operations

Let f be an ACI operation preserving $\mathcal{R}$. Since f is associative and commutative we have

$$f(\langle 0,2,1\rangle, \langle 1,0,2\rangle) = \langle 0,0,1\rangle \text{ or } \langle 0,2,2\rangle.$$

In the first case we get a contradiction considering $f(\langle 0,2,2\rangle, \langle 1,0,2\rangle) = \langle 0,0,2\rangle \notin \mathcal{R}$. For the second case, take $f(\langle 0,0,1\rangle, \langle 1,0,2\rangle) = \langle 0,0,2\rangle \notin \mathcal{R}$

Chapter 7

Para-Primal Algebras are Tractable

7.1 Introduction

In this chapter we continue further the characterization of efficiently solvable instances of the constraint satisfaction problem.

As we discussed in Section 6.1, the family of constraint satisfaction problems obtained when the constraint relations (relations used in the constraints) belong to a finite and fixed set Γ, denoted by CSP(Γ), has a natural formulation as the satisfiability problem over the class of generalized formulas with basis Γ.

Furthermore, the computational complexity of solving CSP(Γ) depends on an algebraic property of Γ: its clone of polymorphisms. This remarkable result started a research project consisting in to identify all the clones that guarantee efficient solvability.

This approach has lead to the identification of several tractable families of problems or basis. The exhaustive list of tractable families known so far in the framework of closure functions is: near-unanimity problems, coset generating problems (see Section 6.3.1) and width 1 problems (Section 6.5).

In this chapter we increase the previous list with a new family of tractable basis that we call *para-primal* basis, incomparable with the previous three classes.

This family of basis is tightly related with a family of algebras that is well known in Universal algebra: para-primal algebras [10]. In this chapter we make an intensive use of algebraic results. So far, the link between constraint satisfaction problem with clones in universal algebra has been used mostly to establish an unifying point of view. Nevertheless, a genuine application of results in clone theory to constraint satisfaction problem is rather rare. In fact, with the exception of the use of Post's classification to give an alternative and simpler proof of Schaefer's dichotomy theorem in [51], the few results in clone theory used in the constraint literature can be easily reformulated and proved in constraint terms. The results of this chapter constitute a genuine application of clone theory to constraint satisfaction problems and a clear evidence of the power of the algebraic-based approach in the study of constraint satisfaction problems. On the other hand, due precisely to the profusion of algebraic concepts and results, some familiarity with algebraic notions would be advisable to follow some sections of the chapter.

The class of para-primal problems has some interesting interpretations. In the previous chapter we shown that given a set of relations Γ, the co-clone generated by Γ, $\langle\Gamma\rangle_r$ is a good representative of the expressivity power of CSP(Γ). In other words: in some technical sense, using constraint relations in Γ it is possible to "simulate" any constraint with constraint relation in $\langle\Gamma\rangle_r$. The binary relations "simulated" by para-primal basis are either: (1) a direct product of two subsets of the domain, or (2) a bijection between two subsets of the domain. In fact, under some additional technical assumptions, the previous property characterizes exactly para-primal basis. Thus, tractability of para-primal algebras can be interpreted in the following way: for any arbitrary basis Γ if the binary relations "simulated" by Γ are "simple enough" then the complete problem is efficiently solvable.

An alternative interpretation is obtained considering it as an extension of the subgroup problems introduced by Feder and Vardi [31]. In a subgroup problem every relation in the basis is a coset of some subgroup of some group power for a fixed group. Feder and Vardi proposed a way to extend the previous definition to include relations others than cosets. More precisely, they introduced a generalization of the concept of coset, called nearsubgroup and they proved that for several families of groups, this extended problem is still efficiently solvable. Para-primal basis can be regarded as a different extension of subgroup problems. Nevertheless, in para-primal algebras the approach relies in combining different subgroup problems rather than extending a unique subgroup problem. In a para-primal algebra, different subgroup problems are combined in such a way that the interaction between them is bounded and the problem can be solved by decomposing the problem in subproblems corresponding to every one of the subgroup problems.

As a consequence of the tractability of para-primal algebras we prove a complete classification for constraint satisfaction problems under the assumption that the basis contains all the permutations. This result generalizes a known previous result due to Cooper et al. [15] in several ways.

7.2 Para-Primal Algebras

In this chapter we assume familiarity with some basic algebraic concepts. See [84] for definitions and results about para-primal algebras.

Let B a subset of A^n $(n \geq 1)$. The symbol $B \preceq B_1 \times \cdots \times B_n$ will be used to designate that $\mathrm{pr}_i B = B_i$ for all $1 \leq i \leq n$. The number $\max\{|B_i| : 1 \leq i \leq n\}$ will be called the *size* of B. Let B and B' subsets of A^n, we denote by $B \approx B'$ the property that B and B' coincide up to the order of their components. The subset B will be termed *directly indecomposable* iff $B \not\approx (\mathrm{pr}_I B) \times (\mathrm{pr}_J B)$ holds for all partitions $\{I, J\}$ of $\{1, \ldots, n\}$.

Furthermore, B is said to be *reduced* iff B is directly indecomposable and no projection $B_{\{i,j\}}$ $(1 \leq i < j \leq n)$ of B is of the form $\pi^\square$ (see Section 5.2) for a bijection $\pi : B_i \to B_j$.

An isomorphism between two subalgebras of an algebra $\mathbf{A} = (A; F)$ is called an *internal isomorphism* of $\mathbf{A}$. Notice that a bijection $\pi : B \to C$ with $B, C \subseteq A$ is an internal isomorphism of $\mathbf{A} = (A; F)$ if and only if $\pi^\square$ is a subuniverse of $\mathbf{A}^2$.

Definition 20 Let A be a finite set and $B \subseteq A^k$ $(k \geq 1)$. A nonvoid subset I of $\{1, \ldots, k\}$ is said to be *B-minimal* iff it is minimal in the sense that the projection $B \to \mathrm{pr}_I B$ is one-to-one.

Definition 21 An algebra $\mathbf{A}$ is called *para-primal* iff it is finite and for every natural number k, every subuniverse B of $\mathbf{A}^k$, and every B-minimal set I, we have $\mathrm{pr}_I B = \prod_{i \in I} B_i$.

As a consequence of the definition we have that if $\mathbf{A}$ is para-primal, then $\mathbf{A}^2$ has no reduced subuniverses. This is precisely the property of para-primal algebras that we will use. Actually, for finite idempotent algebras the previous condition characterizes exactly para-primality.

Theorem 28 *([84],Corollary 4.13) A finite idempotent algebra $\mathbf{A}$ is para-primal if and only if $\mathbf{A}^2$ has no reduced subuniverses.*

In the rest of the chapter we will assume idempotency. This assumption is fully justified because of the following result.

Lemma 30 *([84],Lemma 4.8) A finite algebra is para-primal if and only if its full idempotent reduct is para-primal.*

In consequence:

Lemma 31 *If all idempotent para-primal algebras are tractable, then every para-primal algebra is tractable.*

Proof. Let $\mathbf{A}$ be any para-primal algebra and let F be its full idempotent reduct. Recall that $\mathbf{A}$ is tractable iff $T(\mathbf{A})$ is tractable. By definition, $F \subseteq T(\mathbf{A})$. F is para-primal and idempotent and consequently tractable. Thus, by virtue of Corollary 5, $T(\mathbf{A})$ is tractable. ■

Thus, tractability of idempotent para-primal algebras implies tractability of para-primal algebras in general. The same result could be obtained as a consequence of the classification of the minimal para-primal algebras given later.

7.2.1 Some Examples

We present some examples of para-primal algebras. Para-primality of the following algebras is a direct consequence of the general description of para-primal algebras presented later, but it can be proved directly by showing that they do not exhibit subuniverses of arity 2.

The Discriminator Operation

For a set A, the *discriminator* t on A is defined by

$$t(a,b,c) = \begin{cases} c & \text{if } a = b \\ a & \text{otherwise} \end{cases}$$

We will see later that algebra $(A; t)$ is para-primal. Actually, it is known that finite algebras containing the discriminator operator t among their term operations constitute an important subclass of para-primal algebras, called *quasi-primal* algebras. Quasi-primal algebras satisfy the following alternative characterization: An algebra $\mathbf{A}$ is quasi-primal algebra iff $\mathbf{A}$ is finite and, for every $k \geq 0$, $\mathbf{A}^k$ has no reduced subuniverses.

The term operation $d(x, y, z) = t(x, t(x, y, z), z)$ is the following majority operation:

$$d(a, b, c) = \begin{cases} a & \text{if a=b} \\ c & \text{otherwise} \end{cases}$$

Actually, operation d has been intensively studied (See [84]). It is called *dual discriminator*.

Affine operations on Universes of Prime Order

Let f be an affine operation on a universe A of primer order. Then $(A; f)$ is para-primal. Actually the original idea in the definition of para-primal algebras was to extend quasi-primal algebras in order to contain groups of prime order [10]. As we mentioned in Section 5.6.1, the subuniverses of $(A; f)^n$ are precisely the subsets of A^n which can be expressed as a system of counting functions modulo $|A|$.

Switching Operation

The simplest example of a para-primal algebra not contained in the so far tractable cases is the *switching operation* s on A, defined by:

$$s(a, b, c) = \begin{cases} a & \text{if } b = c \\ b & \text{if } a = c \\ c & \text{otherwise} \end{cases}$$

Algebra $(A; s)$ is para-primal. Notice that restricted to a boolean domain $A = \{0, 1\}$, the switching operation $s(a, b, c)$ is equivalent to the affine operation $(a \oplus b \oplus c)$. For larger domains $|A| = k \geq 3$, algebra $(A; s)$ is not affine. To see it, notice that (A, s) has subuniverses of arity $k - 1$ which does not divide k for $k \geq 3$. Thus, switching operation constitutes a different extension of the affine class in the boolean case to larger domains.

Switching operation will be important in the study of homogeneous algebras.

7.2.2 Tractability of Para-Primal algebras

In this section we will prove that para-primal algebras are tractable.

Theorem 29 *Let* $\mathbf{A} = (A; F)$ *be an idempotent para-primal algebra. Then, for every finite set of relations* Γ *over* A *with* $F = \mathrm{Pol}(\Gamma)$*, the class of problems* $\mathrm{CSP}(\Gamma)$ *is tractable.*

Similarly to the previous tractable classes, we will say that a finite set of relations Γ over a finite set A is para-primal if $(A; \mathrm{Pol}(\Gamma))$ is para-primal.

The proof is splitted in two parts. First, we prove that subuniverses of product powers of para-primal algebras satisfy a certain decomposability property. Then, we derive an algorithm for tractability that uses this decomposability property.

Structure of the Subuniverses

The key notion of the section is that of *isomorphic-decomposable.* First, we introduce some notation.

Let $\mathbf{A} = (A; F)$ be an algebra. Consider the isomorphy relation $\approx$ in the set of subalgebras of $\mathbf{A}$. Clearly, relation $\approx$ is an equivalence relation and, in consequence, it determines a partition of the subalgebras of $\mathbf{A}$.

Let $B \preceq B_1 \times \cdots \times B_n$ be an arbitrary subuniverse of $\mathbf{A}^n$. Thus, the isomorphy relation determines a partition $I_1, \ldots, I_m$ of $\{1, \ldots, n\}$ in the following way: let $\mathbf{B_i} = (B_i; F)$ $(1 \leq i \leq n)$ be the subalgebra of $\mathbf{A}$ with universe B_i. Then $1 \leq i, j \leq n$ belong to the same class iff $\mathbf{B_i} \approx \mathbf{B_j}$. We will say that $I_1, \ldots, I_m$ is the *isomorphic-decomposition associated to* B *with respect to* $\mathbf{A}$.

Relation B is said to be *isomorphic-decomposable* if for all $t \in A^n$,

$$\mathrm{pr}_{I_1} t \in \mathrm{pr}_{I_1} B, \ldots, \mathrm{pr}_{I_m} t \in \mathrm{pr}_{I_m} B \Longrightarrow t \in B.$$

For simplicity, we will assume that I_1 contains the first $|I_1|$ indices in $\{1, \ldots, n\}$ and so on. Thus, B is isomorphic-decomposable iff $B = \mathrm{pr}_{I_1} B \times \cdots \times \mathrm{pr}_{I_m} B_m$.

Lemma 32 *For any algebra* $\mathbf{A}$ *and for any non-isomorphic-decomposable subuniverse* B *of* $\mathbf{A}^n$ *there exists some set of indices* $I \subseteq \{1, \ldots, n\}$ *such that* $\mathrm{pr}_I B$ *is reduced and non-isomorphic-decomposable.*

Proof. We need two intermediate results:

Let $B \preceq B_1 \times \cdots \times B_n$ be a non-isomorphic-decomposable subuniverse of $\mathbf{A}^n$, and let i, j two indices such that $B_{\{i,j\}}$ is $\pi^{\square}$ for some bijection $\pi : B_i \to B_j$. Then $B' = B_{\{1,\ldots,j-1,j+1,\ldots,n\}}$ is non-isomorphic-decomposable.

Proof: Let $I_1, \ldots, I_m$ be isomorphic-decomposition associated to B with respect to $\mathbf{A}$. Clearly i, j belong to the same class, say I_k. By the non-isomorphic decomposability of B there exists $t \notin B$, such that for all $1 \leq l \leq m$, $\mathrm{pr}_{I_l} t \in \mathrm{pr}_{I_l} B$.

Let $t' = \mathrm{pr}_{\{1,\ldots,j-1,j+1,\ldots,n\}} t$. We will see that assignment t' proves the non-isomorphic-decomposability of B'. Clearly, $I_1, \ldots, I_l/\{j\}, \ldots, I_m$ is the isomorphic-decomposition associated to relation B'. By construction we have $\mathrm{pr}_{I_l/\{j\}} t \in \mathrm{pr}_{I_l/\{j\}} B'$. Furthermore, $t' \notin B$, since $\mathrm{pr}_{\{i,j\}} B$ is a bijection and, in consequence, t is the unique valid extension of t'. Thus, t' certificates the non-isomorphic-decomposability of B' and the proof is done.

The second result that we will use is: Let $B \preceq B_1 \times \cdots \times B_n$ be an non-isomorphic-decomposable subuniverse of $\mathbf{A}^n$. If B is directly decomposable, then there exists some proper subset of indices $I \subset \{1, \ldots, n\}$ such that $\mathrm{pr}_I B$ is non-isomorphic-decomposable.

Proof: Let I, J be any partition of $\{1, \ldots, n\}$ such that $B \approx \mathrm{pr}_I B \times \mathrm{pr}_J B$. Clearly B is isomorphic-decomposable whenever both $\mathrm{pr}_I B$ and $\mathrm{pr}_J B$ are isomorphic-decomposable.

The main result is proved by induction in the arity n of B. If $(n = 1)$, then relation B is trivially isomorphic-decomposable. Consider the case $(n > 1)$. Let $B \preceq B_1 \times \cdots \times B_n$ be any non-isomorphic-decomposable subuniverse of $\mathbf{A}^n$. If B is reduced we are done. Otherwise, we should consider two possibilities: (1) B is directly decomposable, in this case there exists some set of indices I with $|I| < n$ such that B_I is non-isomorphic-decomposable. Otherwise (2) there exists two indices $1 \leq i, j \leq n$ such that $\mathrm{pr}_{\{i,j\}} B$ is $\pi\Box$ for some bijection $\pi : B_i \to B_j$. Thus, $\mathrm{pr}_{\{1,\ldots,j-1,j+1,\ldots,n\}} B$ is non-isomorphic-decomposable.

In consequence, in both cases there exists some set of indices I with $|I| < n$ such that $\mathrm{pr}_I B$ is non-isomorphic-decomposable. Then, there exists some subset $I' \subseteq I$, such that $\mathrm{pr}_{I'} B$ is non-isomorphic-decomposable and reduced. ■

Now, we will apply some known results to prove that if an algebra $\mathbf{A}$ is para-primal, then $\mathbf{A}^n$ has no reduced non-isomorphic-decomposable subuniverses. Then, by contradiction, we have the following result: Every subuniverse of $\mathbf{A}^n$ is isomorphic-decomposable.

We study two cases: quasi-primal and non-quasi-primal algebras. The first case is easy, since if $\mathbf{A}$ is a quasi-primal algebra then $\mathbf{A}^n$ does not have any reduced subuniverse at all. If $\mathbf{A}$ is not quasi-primal, then the following result is applicable.

Theorem 30 *([84],Theorem 4.3) Let $\mathbf{A} = (A; F)$ be a finite idempotent algebra. Suppose $\mathbf{A}$ is not quasi-primal, and let $B \preceq B_1 \times \cdots \times B_n$ be a reduced subuniverse of $\mathbf{A}^n$ for some $n \geq 2$. Then either*

a.- $\mathbf{A}^2$ has a reduced subuniverse of the same size as B, or

b.- $\mathbf{B_i} = (B_i; F)$ $(1 \leq i \leq n)$ are all isomorphic affine subalgebras.

See [84] for the definition of affine algebras. The only property of affine algebras that we will use in this chapter is that affine algebras contain an affine operation among their term operations.

As a consequence of the Theorem 30 and Theorem 28 we have that, if $\mathbf{A}$ is para-primal, condition (b) holds and, in consequence, every reduced subuniverse $B \preceq B_1 \times \cdots \times B_n$ is isomorphic-decomposable, since $\mathbf{B_i}$ are isomorphic algebras.

Thus, in both cases (quasi-primal and non-quasi-primal algebras) we can imply that there exists no isomorphic-decomposable reduced subuniverses of $\mathbf{A}^n$, and, by virtue of Lemma 32, every subuniverse of $\mathbf{A}^n$ is isomorphic-decomposable.

Furthermore, we have the following interesting property. Let I_l $(1 \leq l \leq m)$ be any of the equivalence classes given by the isomorphy relation in $B \preceq B_1 \times \cdots \times B_n$. Let I be any set of indices containing exactly one of the representatives of the relation: $i \sim j$ $(i, j \in I_l)$ iff $\mathrm{pr}_{\{i,j\}} B$ is of the form $\pi\Box$ for a bijection $\pi : B_i \to B_j$. Notice that if $\mathrm{pr}_I B \neq \prod_{i \in I} B_i$, then there exists some $J \subseteq I$ such that $\mathrm{pr}_J B$ is reduced. Then, the previous theorem implies that $\mathrm{pr}_J B$ (and consequently $\mathrm{pr}_{I_l} B$) is preserved by an affine operation. Otherwise, $B_I = \prod_{i \in I} B_i$. It can be easily verified that, in this case, discriminator t preserves $\mathrm{pr}_{I_l} B$.

Summarizing,

Theorem 31 *Let $\mathbf{A}$ be a para-primal algebra and let B be a subuniverse of $\mathbf{A}^n$, then B is isomorphic-decomposable. Furthermore, let $I_1, \ldots, I_m$ be the isomorphic-decomposition associated to B in $\mathbf{A}$, then for every $1 \leq l \leq m$, $\mathrm{pr}_{I_l} B$ is preserved by an affine operation or a discriminator operation.*

Algorithm

The underlying idea of the algorithm is that using the decomposability property shown in the previous section, it is possible, after an appropriate trans-formation of the problem, to divide the problem in subproblems. Every subproblem falls in one of the following already known tractable classes: affine problems and dual discriminator problems.

Let $\mathcal{P}$ be a constraint instance in CSP(Γ) over the variables $x_1, \ldots, x_n$ and with domain A. Let k be the largest arity among the relations in Γ and let $\mathcal{P}_1$ be the problem obtained enforcing $(1, k)$-consistency to $\mathcal{P}$. For every variable x_i $(1 \leq i \leq n)$, let D_{x_i} be the set of values that can be assigned to x_i, i.e., the set of solutions of the problem restricted to x_i.

Replace every constraint $\langle (x_{i_1} \ldots, x_{i_k}), R\rangle$ in $\mathcal{P}_1$ by the new constraint $\langle (x_{i_1} \ldots, x_{i_k}), R \cap (D_{x_{i_1}} \times \cdots \times D_{x_{i_k}})\rangle$ obtaining a new problem $\mathcal{Q}$ with identical set of solutions. Furthermore, $\mathcal{Q}$ satisfies the following property: Let $\langle (x_{i_1}, \ldots, x_{i_k}), R\rangle$ be any constraint in $\mathcal{Q}$, then $R \preceq D_{x_{i_1}} \times \cdots \times D_{x_{i_k}}$.

Consider the isomorphy relation over $\mathbf{D_{x_1}}, \ldots, \mathbf{D_{x_n}}$, as defined above. Consider any constraint $C = \langle s, R\rangle$ in $\mathcal{Q}$. As a consequence of the construction of C and the remark before Definition 19 we have that $R \in \langle \Gamma \rangle_r$. Thus R is isomorphic-decomposable. In consequence, it is possible to replace the constraint C by the new constraints $\langle s_1, R_1\rangle, \ldots, \langle s_m, R_m\rangle$ obtained by dividing R in its components. Thus, we can assume that for every constraint $C = \langle (x_{i_1}, \ldots, x_{i_k}), R\rangle$ in $\mathcal{Q}$ we have that $\mathbf{D_{x_{i_1}}}, \ldots, \mathbf{D_{x_{i_k}}}$ are isomorphic.

Since the scope of every constraint C only involves variables belonging to the same isomorphy class, we can split problem $\mathcal{Q}$ in subproblems $\mathcal{Q}_1, \ldots, \mathcal{Q}_r$. Each problem $\mathcal{Q}_i$ $(1 \leq i \leq r)$ containing the variables of an isomorphy class in $\mathbf{D_{x_1}}, \ldots, \mathbf{D_{x_m}}$.

Consider a subproblem $\mathcal{Q}_i$ $(1 \leq i \leq r)$. If problem $\mathcal{Q}_i$ corresponds to an affine isomorphy class, then every constraint relation occurring in $\mathcal{Q}_i$ is preserved by the corresponding affine operation and, consequently solvable. Otherwise, every constraint relation in $\mathcal{Q}_i$ is preserved by the discriminator operation t and, consequently, by the dual discriminator d. In this case, $\mathcal{Q}_i$ is solvable using the 2-consistency algorithm (see Section 6.3.1).

7.2.3 General Description of Para-Primal Algebras

The structure of para-primal algebras has been intensively studied [10, 11, 77] (see [84] for an overview). It is known that para-primal algebras constitute an order filter in the clone lattice. For a finite universe A, up to term equivalence, there are only finitely many minimal para-primal algebras which were identified by Quackenbush [77].

Let A be a finite set, and $A_1, \ldots, A_k$ be a family of subsets of A with $|A_i \cap A_j| \leq 1$ $(1 \leq i < j \leq k)$ and $|A_i| = q_i$, a prime power, for all $1 \leq i \leq k$.

Let us fix a generating element c_i of the multiplicative group of each finite galois field $\mathrm{GF}(q_i)$ so that $c_i = c_j$ whenever $q_i = q_j$ $(1 \leq i, j \leq k)$. Consider a vector space ${}_{\mathrm{GF}(q_i)}\underline{A_i} = (A_i; +_i, -_i, \mathrm{GF}(q_i))$ for every $1 \leq i \leq k$, and define a ternary operation p on A by,

$$p(a,b,c) = \begin{cases} a -_i b +_i c & \text{if } a,b,c \in A_i \text{ for some } 1 \leq i \leq k \\ t(a,b,c) & \text{otherwise} \end{cases}$$

and a binary operation g on A by

$$g(a,b) = \begin{cases} c_i a +_i (1 - c_i) b & \text{if } a,b \in A_i \text{ for some } 1 \leq i \leq k \\ a & \text{otherwise} \end{cases}$$

An algebra $(A; p, g)$ as construced above is called a **MPP** (Minimal Para-Primal) algebra.

Algebra **MPP** is exactly the minimal para-primal algebra with nonsingleton affine subuniverses $A_1, \ldots, A_k$.

Theorem 32 *([84], Theorem 4.15) The algebra $(A; p, g)$ constructed above is para-primal, and has no proper para-primal reduct in which the nonsingleton affine subuniverses are exactly $A_1, \ldots, A_k$. Moreover, every para-primal algebra $(A; F)$ with nonsingleton affine subuniverses $A_1, \ldots, A_k$ has a reduct of the form $(A; p, g)$.*

The examples in Section 7.2.1 follow directly from the previous description. The algebra $(A; t)$ coincides with (A, p, g) taking $k = 0$, since g is the first projection and $p = t$. Taking A with prime size and $k = 1$ with $A_1 = A$ we have that

$$g(a,b) = \overbrace{f \ldots f}^{c_1 - 1}(a,b,a), b, a) \ldots), b, a)$$

and $p(a,b,c) = f(a,b,c)$. Finally, for $|A| = n$, consider the family of subsets $A_1 \ldots, A_{\binom{n}{2}}$ given by all the 2-element subsets of A. Clearly g is the first projection and $p(a,b,c) = s(a,b,c)$.

Another interesting consequence of the previous theorem is that given a finite set of relations Γ over A it is possible to decide whether the algebra $(A; \mathrm{Pol}(\Gamma))$ is para-primal, by testing the closure property for all possible pairs of operations p and g.

Thus, we have

Theorem 33 *The problem of deciding whether a finite set of relations Γ is para-primal is decidable.*

7.2.4 The Class of Para-Primal Algebras

As we have discussed in Chapter 6, every clone that contains a tractable clone is tractable. In consequence, all the families of tractable clones are identified by their minimal elements. Accordingly, we characterize the family of paraprimal algebras with the minimal elements.

Theorem 34 *Let Γ be a set of relations over A such that Γ is preserved by two operations p and g, as defined above. Then* $\mathrm{CSP}(\Gamma)$ *is tractable.*

Let us place the class of para-primal algebras with respect to the classes already known to be tractable. Recall that, so far, there are three main families of tractable classes, identified by Feder and Vardi [31]. These families are width 1 problems, bounded strict width problems, and subgroup problems (see Section 6.3.1).

The class of para-primal algebras contains some problems included in Feder and Vardi's classification. Among the examples presented in Section 7.2.1, the discriminator operation and affine operations belong to the classes of bounded strict and subgroup problems respectively.

Nevertheless, some classes such as switching operation cannot be included in any of the three families. We will give only a brief idea of the proof. First, swithing operation are not included in width 1 and bounded width 1 problems because, restricted to a 2-element domain, switching operation is equivalent to affine problems wich are known not to be in the previous classes [31]. Furthermore, if the domain D contains at least three elements, then swithing operation is not included in the class of subgroup problems. To see it recall that any relation in a subgroup problem is a coset of some subgroup of a group power for some fixed group, and therefore, they should satisfy some restriction in the number of elements of the relation, as discussed in Section 5.6.1. More precisely the number of elements of a k-ary relation should divide $|D|^k$. We get a contradiction with the fact that any unary relation is preserved by the switching operation. Therefore, the class of para-primal algebras constitutes a new tractable class.

The class of problems associated to para-primal algebras can be regarded as a combination of some affine algebras, which are particular cases of subgroup problems. Feder and Vardi [31] introduced another way to extend the class of subgroup problems. The approach introduced in [31] is based in adding some non-subgroup constraints rather that in combining different subgroup problems.

7.3 An Application: Dichotomy under the Availability of Permutation Relations

A dichotomic classification of CSP(Γ) is a major open question. In fact, the mere existence of such a dichotomy is an interesting theoretical question. It is known that restricting the problem in several ways we have a dichotomy [81, 41, 15]. In this section we extend a known dichotomy classification of constraint satisfaction problems due to Cooper et al. [15]. The problem addressed in [15] is to classify the computational complexity of constraint satisfaction problems in the basis: (1) contains only binary relations, (2) includes all the unary relations, and (3) includes all the permutation relations, ie. relations of the form $\pi^\square$ where $\pi : A \to A$ is a permutation. In [15], it is proved that the only family of tractable basis under the above-mentioned restriction is the class of the so-called 0/1/all relations which in our terminology can be defined as the binary relations preserved by the dual discriminator operation d.

In this section we use the tractability of switching operation and some known results about the classification of homogeneous algebras to extend the previous dichotomy by dropping restrictions (1) and (2). Let A be a finite set. A *permutation relation* on A is any relation of the form $\pi^\square$ where $\pi : A \to A$ is a permutation. We will study the problems given by basis of relations containing all the permutation relations on the domain of the basis. Let Γ be a finite set of relations on A. Then

$$\mathrm{CSP}^*(\Gamma) = \mathrm{CSP}(\Gamma \cup \{R : R \text{ permutation relation on } A\})$$

Homogeneous Algebras

The key observation of this section is that for every permutation $\pi : A \to A$, $\pi^\square$ is a subuniverse of $\mathbf{A}$ if and only if π is an automorphism of $\mathbf{A}$. Thus, we can restrict to algebras where every permutation is an automorphism.

Definition 22 An algebra $\mathbf{A} = (A; F)$ is called *homogeneous* iff every permutation of A is an automorphism of $\mathbf{A}$. An operation f on A is homogeneous if $(A; f)$ is homogeneous.

Example 4 Let A be a finite set, $|A| = n$. The following operations on A are homogeneous:

- the switching operation s;
- the discriminator operation t;
- the dual discriminator operation d;
- the k-ary *near-projection* l_k $(3 \leq k \leq n)$ defined by

$$l_k(a_1, \ldots, a_k) = \begin{cases} a_1 & \text{if } |\{a_1, \ldots, a_k\}| < k \\ a_k & \text{otherwise;} \end{cases}$$

- the $(n-1)$-ary operation r_n defined by

$$r_n(a_1, \ldots, a_{n-1}) = \begin{cases} a_1 & \text{if } |\{a_1, \ldots, a_{n-1}\}| < n-1 \\ a_n \text{ with } \{a_n\} = A/\{a_1, \ldots, a_{n-1}\} & \text{otherwise;} \end{cases}$$

  for $n = 2$, r_2 is the unique nonidentity permutation on A, while for $n = 3$, $r_3(x, y) = 2x + 2y$ for arbitrary group $(A, +)$;

- the $(n-1)$-ary operation d_n defined by

$$d_n(a_1, \ldots, a_{n-1}) = \begin{cases} d(a_1, a_2, a_3) & \text{if } |\{a_1, \ldots, a_{n-1}\}| < n-1 \\ a_n \text{ with } \{a_n\} = A/\{a_1, \ldots, a_{n-1}\} & \text{otherwise;} \end{cases}$$

  provided $n \geq 4$.

- the operation $x + y + z$ if $(A, +)$ is a 4-element group of exponent 2.

Homogeneous algebras were first investigated by Marczewski [63] and were finally completely classified by Marchenkov [62]. Here we give a description of Marchenkov's classification (See [84] for a detailed presentation of homogeneous algebras). There exists some "exceptional" homogeneous algebras on small sets ($|A| \leq 5$). Thus, they are presented separately. The lattice of clones of homogeneous algebras on a set A of $n(\geq 5)$ elements is depicted if Figure 7.1. The lattice of clones of homogeneous algebras on a 4-element, 3-element, and 2-element set are depicted in Figure 7.2. The notation is as follows:

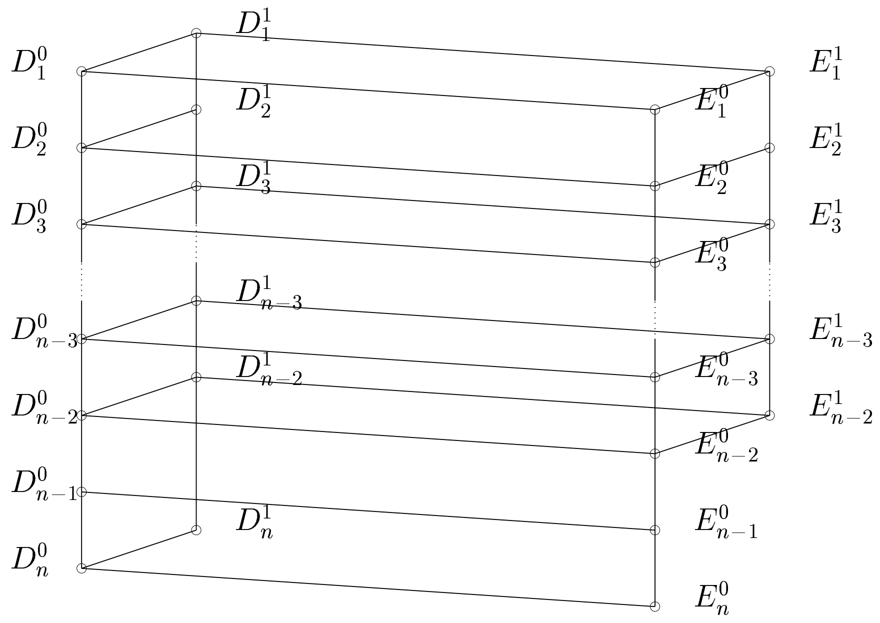

Figure 7.1: Lattice of homogeneous algebras with size ≥ 5

- $D_1^0 = \langle t \rangle_o$; $D_n^0 = \langle d \rangle_o$;
- $D_i^0 = \langle d, l_{i+1} \rangle_o$ for $2 \leq i \leq n-1$;
- $D_1^1 = \langle t, r_n \rangle_o$; $D_n^1 = \langle d_n \rangle_o$;
- $D_i^1 = \langle d, l_{i+1}, r_n \rangle_o$ for $2 \leq i \leq n-2$;
- $E_1^0 = \langle s \rangle_o$; $E_n^0 = \mathcal{I}$
- $E_i^0 = \langle l_{i+1} \rangle_o$ for $2 \leq i \leq n-1$;
- $E_1^1 = \langle s, r_n \rangle_o$; $E_{n-2}^1 = \langle r_n \rangle_o$;
- $E_i^1 = \langle l_{i+1}, r_n \rangle_o$ for $2 \leq i \leq n-3$;

Since the classification of homogeneous algebras is complete we can study tractability for every element in the classification individually. Let us consider first some tractable clones:

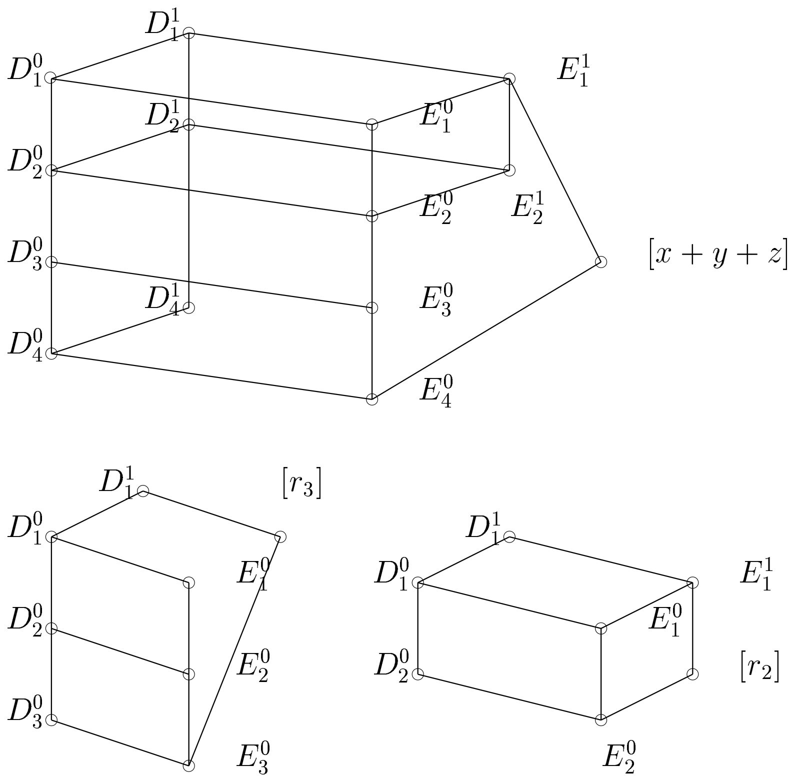

Figure 7.2: Lattice of homogeneous algebras with size ≤ 4

- The dual discriminator d is a majority operation and, hence, tractable (Section 6.3.1). In consequence, D_i^j $(1 \leq i, \;\; j \in \{0,1\})$ is tractable, since d is contained in them.

- The switching operation s is tractable and therefore E_1^j $j \in \{0,1\}$ are tractable.

Now consider the remaining clones. Fix two arbitrary values $a, b \in A$. Notice that operations l_k $(3 \leq k \leq n)$ and r_n $(n \geq 4)$ preserve all the relations over any 2-element subdomain $\{a, b\}$, and in particular the NOT-

ALL-EQUAL with set of tuples

$$\{ \quad \begin{array}{l} \langle a, a, b \rangle \\ \langle a, b, a \rangle \\ \langle b, a, a \rangle \\ \langle b, b, a \rangle \\ \langle b, a, b \rangle \\ \langle b, b, a \rangle \quad \} \end{array}$$

which is known to be NP-complete [81]. In consequence, E_i^0 $(2 \leq i \leq n)$ and E_i^1 $(2 \leq i \leq n-2)$ are NP-complete.

Thus, homogeneous algebras with $|A| = n \geq 5$ are completely classified, being the tractable clones the ones containing the switching operation s and the dual discriminator d.

For $n = 4$ we have to study the special case of the clone generated by $(x + y + z)$ where $(A, +)$ is a group of exponent 2. Operation $(x + y + z)$ is affine and, in consequence, the clone is tractable. Thus, for $n = 4$ the only tractable classes are the clones containing s, d or $(x + y + z)$.

For $n = 3$ we have to consider the particular case $\langle r_3 \rangle_o$. Clearly, operation r_3 is tractable since $r_3(r_3(x, y), r_3(y, x)) = 2(2(x+y)+2(y+z)) = 4x+8y+4z = x + 2y + z = x - y + z$. Thus, the tractable classes for $n = 3$ are the clones containing s, d, and r_3.

For $n = 2$, we have to consider operation r_2, which is the non-identity permutation. Clearly r_2 preserves the NOT-ALL-EQUAL relation and therefore clone $\langle r_2 \rangle_o$ is NP-complete. Thus summarizing the tractable classes for $n = 2$ are the clones containing d and s.

The results in this section can be summarized in the following theorem.

Theorem 35 *Let Γ be a set of relations over A. Then* $\mathrm{CSP}^*(\Gamma)$ *is tractable iff* $\mathrm{Pol}(\Gamma)$ *contains the dual discriminator d; or the switching operation s; or an affine function. More precisely,* $\mathrm{CSP}^*(\Gamma)$ *is tractable if and only if*

- $\{s, d\} \cap \mathrm{Pol}(\Gamma) \neq \emptyset$ *if* $|A| \notin \{3, 4\}$.
- $\{s, d, x+y+z\} \cap \mathrm{Pol}(\Gamma) \neq \emptyset$, *where* $(A, +)$ *is a group of exponent* 2, *if* $|A| = 4$.
- $\{s, d, r_3\} \cap \mathrm{Pol}(\Gamma) \neq \emptyset$, *if* $|A| = 3$.

■

Chapter 8

Learnability of Generalized Formulas

8.1 Introduction

In Chapter 3 we studied the learnability of generalized quantified formulas over the boolean basis. We proved that for every finite boolean basis Γ the class of generalized quantified boolean formulas over Γ is either polynomially learnable in the restricted model of exact learning with equivalence queries, or not polynomially predictable in the most powerful model of PAC-prediction even if we add the availability of asking membership queries. Furthermore, we gave a complete classification of the learnable cases.

The main goal of this chapter 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 variable, rather than boolean variables only.

As a main intermediate tool for our study we introduce a link with *clones* in Universal Algebra. This approach has been introduced in Chapter 6 and it has been successful in the study of the Constraint Satisfaction Problem (CSP). 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 show that the coset generating (CG), near-unanimity (NU) and para-primal basis are efficiently learnable. This list contains, as a particular case, all 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 basis with a possibly infinite number of relations. For the boolean domain, the clone lattice is simpler and it has been completely characterized (Section 5.7).

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.

8.2 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 whith 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 algorithm.

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

Definition 23 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 8.2 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 [86] called the *Closure Algorithm*: the output of this algorithm is always $\langle H \rangle_{\mathcal{C}}$ where H contains all the positive samples [69]. Algorithm **GS** is a direct adaptation of the Closure Algorithm. Consequently, it is possible to extend the results in this paper to nested differences of intersection-closed classes using an approach similar to [43].

It is possible to convert algorithm **GS** to a proper algorithm finding, for

```
Algorithm GS
H = ∅,
while EQ(⟨H⟩_C)= 'no' do
        Let c be the counterexample
        H = H ∪ {c}
endwhile
return ⟨H⟩_C.
```

Figure 8.1: Generating Set Algorithm (**GS**)

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 24 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 class. We do not consider the size of the representation class. This choice makes the analysis simpler, since we obtain independence from the particular representation class. Furthermore, as we will see later (Section 8.4), 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 36 *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.

8.3 Learning Subuniverses

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

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 25 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 $\mathcal{C}_{(D;f)}$.

In the remainder of this section we will see that for some families of algebras, namely, coset generating algebras, near-unanimity algebras, and para-primal algebras, the class of subuniverses of the direct power of these algebras is polynomially bounded. In consequence, every class of quantified formulas with a basis preserved by an algebra in one of these families is polynomially learnable using algorithm **GS**.

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 $F = \mathrm{Pol}(\Gamma)$ 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. Recall that 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 33 *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 37 *Let $\mathcal{F}$ be a class of formulas over a finite domain D and let F be the clone 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 33, and the first part of Theorem 36. To show condition (b) just consider that both classes of concepts are intersection-closed and apply the second part of Theorem 36. 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. ∎

8.3.1 Coset Generating Operations

As we mentioned in Section 5.6.1, coset generating operations over a domain D of prime size have a rather peculiar structure. It is known that the subuniverses of its direct powers are exactly the subsets of D^n that can be expressed as a system of counting functions modulo $|D|$. The learnability of these formulas has already been shown in [9] using a similar strategy. The next result generalizes this result to arbitrary CG operations.

Theorem 38 *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* $\mathcal{C}_{(D;\varphi)}$.

The proof contains mainly two results. We have to show that algebras containing CG operations are polynomially bounded and polynomially evaluable. First we prove that $(D;\varphi)$ is polynomially bounded:

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. ∎

The proof of the polynomial-time evaluability mimics the proof of Theorem 32 in [31].

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 [44][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

[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.

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*. ∎

8.3.2 Near-unanimity Operations

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

Theorem 39 *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* $\exists\forall\text{-}\mathrm{Form}_C([\mathcal{F}])$ *is polynomially learnable with proper equivalence queries.*

The proof of the theorem is split into several results.

Lemma 34 *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 19. 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 35 *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 $\{1,2,\dots,n\}$ with $|I| \le r-1$, 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 36 *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 $I = \{i_1,\dots,i_{r-1}\}$ 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.

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

■

Corollary 6 *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:* $\exists$-$\mathrm{Form}([\mathcal{F}])$, $\exists\forall$-$\mathrm{Form}([\mathcal{F}])$, $\exists$-$\mathrm{Form}_C([\mathcal{F}])$, *and* $\exists\forall$-$\mathrm{Form}_C([\mathcal{F}])$.

8.3.3 Para-Primal Algebras

As we showed in the previous chapter (Theorem 31), if $\mathbf{D} = (D; F)$ is a para-primal algebra, then every subuniverse B of $\mathbf{D}^n$ has a rather particular structure. More precisely, it is known that B is isomorphic-decomposable and that every element of the decomposition is preserved by an affine operation or by the discriminator operation t.

Using the previous characterization it is direct to shown that $\mathbf{D}$ is polynomially bounded.

Theorem 40 *Every para-primal algebra is polynomially bounded.*

Proof. Let $\mathbf{D}$ be a para-primal algebra. Let $u = (x_0, \ldots, x_m)$ be any additive sequence over $\mathbf{D}^n$. For every $j \in \{0, \ldots, m+1\}$ we define B^j as the n-ary relation given by $\langle x_0, \ldots, x_{j-1} \rangle_{\mathcal{C}_\mathbf{D}}$. Let $I = \{p, p+1, \ldots, q\}$ be a sequence of consecutive integers. We will say that I is a non-increasing sequence if there exists $D_1, \ldots, D_n \subseteq D$ such that for all $j \in I$ and for all $1 \leq i \leq n$, $\mathrm{pr}_i B^j = D_i$.

First, we calculate a polynomial bound for $|I|$. Since, $\mathbf{D}$ is para-primal $B^p, \ldots, B^q$ are isomorphic-decomposable. Furthermore, $B^p, \ldots, B^q$ have the same isomorphy classes $I_1, \ldots, I_l$. Thus, we can divide every $B^j \ \ (j \in I)$ in its components $B^j_r = \mathrm{pr}_{I_r} B^j \ \ (1 \leq r \leq l)$.

Consider any fixed $I_r \quad (1 \leq r \leq l)$ with $|I_r| = s$. If I_r corresponds to an affine subalgebra $\mathbf{D_i}$ of $\mathbf{D}$ then every subuniverse in the sequence $B^p_r, B^{p+1}_r, \ldots, B^q_r$ is a coset of the product group $(D_i, \cdot, ^{-1}, 1)^s$ for some group $(D_i; \cdot, 1, ^{-1}, 1)$. Therefore every subsequence containing non-repeated elements has a size bounded by a polynomial s. Otherwise $\mathbf{D_i}$ is not affine, then by Theorem 31, every subuniverse $B \preceq D_i^m$ is preserved by the discriminator t and, in consequence, by the dual discriminator d. Every subuniverse in the sequence $B^p_r, B^{p+1}_r, \ldots, B^q_r$ is 2-decomposable. In consequence, every subsequence of universes containing non-repeated subuniverses has a size bounded by a polynomial s. Thus, in both cases there is a polynomial, say p_r, in s that bounds the size of I_r.

We know that $B^p \subset B^{p+1} \subset \cdots \subset B^q$ where the inclusion is strict. This implies that every one of the previous inclusions are proper for at least one of the components $I_1, \ldots, I_l$. Thus, we get the following bound for the size of I

$$|I| \leq \sum_{1 \leq r \leq l} p_r(|I_l|),$$

which can be bounded by a polynomial q in n.

Thus divide $\{0,\ldots,m\}$ in maximal non-increasing sequences. By the previous result the size of every one of such sequences is bounded by a polynomial q in n. Furthermore, we have at most $n|D|$ of such sequences. Thus, m is bounded by a polynomial. ∎

Now, we prove the polynomial-time evaluability.

Theorem 41 *Every para-primal algebra is polynomial-time evaluable.*

Proof. Let $\mathbf{D}$ be a para-primal algebra. Let H be a set of tuples on D^n and let x a tuple on D^n. Let $\langle H\rangle_{\mathcal{C}_\mathbf{D}} \preceq D_1 \times \cdots \times D_n$. Clearly, D_i can be calculated efficiently. Consider the isomorphy-relation in $\mathbf{D_i}$ $(1 \leq i \leq n)$ with classes $I_1,\ldots,I_l$. Thus, $x \in \langle H\rangle_{\mathcal{C}_\mathbf{D}}$ iff $\mathrm{pr}_{I_r} x \in \mathrm{pr}_{I_r}\langle H\rangle_{\mathcal{C}_\mathbf{D}}$ for all $1 \leq r \leq l$. Furthermore, every one of this subproblems is solvable efficiently: If I_l corresponds to an affine subalgebra $\mathbf{D_i}$ then Theorem 38 is applicable. Otherwise, we know that $\mathrm{pr}_{I_r}\langle H\rangle_{\mathcal{C}_\mathbf{D}}$ is 2-decomposable and in consequence, Lemma 35 is applicable. ∎

Since every para-primal relation is preserved by a **MPP** algebra (see Section 7.2.3), we can summarize our results about para-primal algebras in the following theorem.

Theorem 42 *Let $(D;p,g)$ be a* **MPP** *algebra and let $\mathcal{F}$ be a set of formulas closed under p and g, then the class $\mathcal{C}_{(D;p,g)}$ is polynomial-time evaluable and polynomially learnable with equivalence queries. Furthermore, the class $\exists\forall\text{-}\mathrm{Form}_C([\mathcal{F}])$ is polynomially learnable with equivalence queries in $\mathcal{C}_{(D;p,g)}$.*

8.4 Non-learnability Results

In the previous sections 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 some classes of quantified formulas and closure operations is even tighter: the closure operations of a quantified formula determine the learnability complexity.

The proof follows the lines of Theorem 22. The proof depends on several results:

Theorem 43 *For every set of relations, Γ, over a finite set D, we have:*

1. $\exists\text{- Rel}(\Gamma \cup \{\text{eq}_D\}) = \text{Inv}(\text{Pol}(\Gamma))$

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

Proof. Item **(1)** is immediate from Theorem 20 and Theorem 15.

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

$$\exists\text{- Rel}_C(\Gamma \cup \{\text{eq}_D\}) = \exists\text{- Rel}(\Gamma \cup \{[x = 0], \ldots, [x = (|D| - 1)]\} \cup \{\text{eq}_D\}).$$

To prove it, notice that for every $0 \leq i \leq |D|-1$, $[x = i]$ is obtained replacing y by i in the identity eq_D. Thus

$$\exists\text{- Rel}_C(\Gamma \cup \{\text{eq}_D\}) \supseteq \exists\text{- Rel}(\Gamma \cup \{[x = 0], \ldots, [x = (|D| - 1)]\} \cup \{\text{eq}_D\}).$$

For the converse consider any formula Φ in $\exists\text{- Form}_C(\Gamma \cup \{\text{eq}_D\})$. Consider the formula Ψ in $\exists\text{- Form}(\Gamma \cup \{[x = 0], \ldots, [x = (|D| - 1)]\} \cup \{\text{eq}_D\})$ given by

$$\Psi = \exists x_0, \ldots, x_{|D|-1} \Phi' \wedge (x_0 = 0) \wedge (x_1 = 1) \wedge \cdots \wedge (x_{|D|-1} = |D| - 1),$$

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

$$\exists\text{- Rel}_C(\Gamma \cup \{\text{eq}_D\}) \subseteq \exists\text{- Rel}(\Gamma \cup \{[x = 0], \ldots, [x = (|D| - 1)]\} \cup \{\text{eq}_D\}).$$

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

$$\begin{aligned} &\exists\text{- Rel}(\Gamma \cup \{[x = 0], [x = 1], \ldots, [x = (|D| - 1)]\} \cup \{\text{eq}_D\}) = \\ &\qquad = \text{Inv}(\text{Pol}(\Gamma \cup \{[x = 0], [x = 1], \ldots, [x = (|D| - 1)]\})) \end{aligned}$$

Finally, notice that $\{[x = 0], [x = 1], \ldots, [x = (|D| - 1)]\}$ is preserved by exactly all the idempotent operations over D. Thus

$$\mathrm{Idem}(\mathrm{Pol}(\Gamma)) = \mathrm{Pol}(\Gamma \cup \{[x = 0], [x = 1], \ldots, [x = (|D| - 1)]\}).$$

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

This result shows that the clone of polymorphisms of a basis Γ characterizes the expressivity power of the class of formulas $\exists\text{-}\mathrm{Form}(\Gamma \cup \{\mathrm{eq}_D\})$. The next result (Lemma 37) shows that in many cases this clone also determines the learning complexity fo this class in the PAC-prediction with membership queries learning model [2].

Lemma 37 *Let Γ and Γ_0 be sets of relations over a finite set D where Γ_0 is finite. Then:*

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

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

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

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

Every relation R in Γ_0 belongs to $\exists\text{-}\mathrm{Form}(\Gamma \cup \{\mathrm{eq}_D\})$ by Theorem 43. Then, given any formula Φ in $\exists\text{-}\mathrm{Form}(\Gamma_0)$, it is possible to replace every occurrence of a relation R in Γ_0 by a formula Ψ_R in $\exists\text{-}\mathrm{Form}(\Gamma \cup \{\mathrm{eq}_D\})$ 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 $\exists\text{-}\mathrm{Form}(\Gamma_0)$ an associated formula in $\exists\text{-}\mathrm{Form}(\Gamma \cup \{\mathrm{eq}_D\})$ constructed as described above and let f and h identity

[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 the main result of this section in which this assumption is used. Furthermore, up to the present moment, there is not known counterexample showing that Lemma 37 is false without this new additional assumption. Thus, the study of the necessity of the finiteness assumption is an interesting open question.

functions. Then, f, g, and h define a pwm-reduction from $\exists\text{-}\mathrm{Form}(\Gamma_0)$ to $\exists\text{-}\mathrm{Form}(\Gamma \cup \{\mathrm{eq}_D)\}$ ∎

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

Lemma 38 *Let Γ be a set of boolean relations over a finite domain D.*

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

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

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

Proof. We prove here item **1**. Proofs of items **2** and **3** are analogous. We give an algorithm to learn $\exists\text{-}\mathrm{Form}(\Gamma \cup \{\mathrm{eq}_D\})$ which uses the algorithm for learning $\exists\text{-}\mathrm{Form}(\Gamma)$ 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 44 *Let Γ and Γ_0 be sets of relations over a finite set D where Γ_0 is finite. Then:*

1. *If $\mathrm{Pol}(\Gamma) \subseteq \mathrm{Pol}(\Gamma_0)$, and $\exists\text{-}\mathrm{Form}(\Gamma)$ is polynomially predictable with membership queries so is $\exists\text{-}\mathrm{Form}(\Gamma_0)$.*

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

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

Proof. Putting the previous results together, Theorem 44 is rather immediate. We only show item **1**. Items **2** and **3** are analogous.

By Lemma 38, $\exists\text{-}\mathrm{Form}(\Gamma \cup \mathrm{eq}_D)$ is polynomially predictable with membership queries. As $\mathrm{Pol}(\Gamma) \subseteq \mathrm{Pol}(\Gamma_0)$, then $\exists\text{-}\mathrm{Form}(\Gamma_0)$ is also polynomially predictable with membership queries. ■

Theorem 44 allows us to classify the complexity of learning any set of quantified formulas over a finite basis, based on the corresponding clone of polymorphisms of that basis.

Theorem 44 also holds for some other learning models as for example: exact learning with equivalence queries, PAC-learning and 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 chapter was to extend the dichotomic classification for the learnability of boolean formulas (Theorem 4) to larger domains. Unfortunately, the 2-element domain is rather particular and whereas the clone lattice for the boolean domain is countable and it was fully identified by Post (Section 5.7), much less is known about the clone lattice for larger domains. In fact, as it has been mentioned in Chapter 5, the clone lattice for $|D| \geq 3$, contains uncountably many clones. 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, plain clones and homogeneous clones we obtain a dichotomy.

8.4.1 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 $\mathcal{C}$.

Learnability of minimal clones is completely classified.

Theorem 45 *(Dichotomy Theorem for minimal clones) Let* Γ *be a set of relations, such that* $\mathrm{Pol}(\Gamma)$ *is minimal. Then, if* $\mathrm{Pol}(\Gamma)$ *contains a majority 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.*

Recall (Corollary 3) that if a set of boolean relations Γ contains $[\overline{x}\vee\overline{y}\vee z]$ or $[x\vee y\vee\overline{z}]$, then $\exists$- $\mathrm{Form}_C(\Gamma)$ is not polynomially predictable with membership queries under cryptographic assumptions. This result will be of intensive use in the rest of the chapter.

The minimal clones are generated by one of the following operations [80, 84]:

1. an idempotent binary operation (which is not a projection); or
2. a majority operation (Section 5.6.2); or
3. an affine operation (Section 5.6.1); or
4. a semiprojection (Section 6.6.3); or
5. a nonidentical unary operation.

The operations in cases **(2)** and **(3)** are particular cases of near-unanimity and coset generating operations respectively, so they are learnable, as shown above. Case **(4)** is also easy to study, since clearly all relations over two values are closed under any semiprojection, in particular so is $[x \vee y \vee \bar{z}]$. Non-learnability of this class follows from application of Corollary 3. Case **(5)** is simple because nonidentical unary operations are not idempotent. Since by Theorem 44, 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 the projections and, in particular so is $[x \vee y \vee \bar{z}]$.

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.

Theorem 46 *[85] 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 grupoids 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 Corollary 3 they are not learnable. This completes the proof. ■

8.4.2 Plain Algebras

An algebra is called *plain* iff it is simple (see [64] for example) and has no nonsingleton proper subalgebras.

The learnability of plain algebras is completely classified.

Theorem 47 *(Dichotomy Theorem for plain clones) 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 (Chapter 3).

In order to establish this result we shall make use the fact that plain idempotent algebras of universes with cardinality larger than two are very restricted, actually they fall in three types identified by Szendrei (see Theorem 2.1 in [85]). First we need some notation.

For a permutation group G acting on D we will denote by $\mathcal{I}_D(G)$ 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 48 *[85] Every plain idempotent algebra* $(D; F)$ *with* $|D| \geq 3$ *is locally term equivalent to one of the following algebras:*

1. *$(D; \mathcal{I}_D(G))$ 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. *$(D; \mathcal{I}_D(G) \cap \mathcal{F}_k^0)$ 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 7 *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 48. In case **1**, $\mathbf{D}$ is locally equivalent to $(D; \mathcal{I}_D(G))$ 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.

As we have seen in Section 7.2.2, every permutation is preserved by the dual discriminator operation d. In consequence d 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 $\mathbf{D}$ (recall that the full idempotent reduct of an algebra $\mathbf{D}$ is an algebra whose operations are the idempotent term operations of $\mathbf{D}$).

In case **3**, $\mathbf{D}$ is locally equivalent to $(D; \mathcal{I}_D(G) \cap \mathcal{F}_k^0)$ 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 $\mathcal{I}_D(G)$.

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 completely classified (this is precisely the content of Theorem 4). It is easy to verify the the tractable cases , namely, bijunctive, affine, weakly monotone and weakly antimonotone correspond to the basis preserved by a near-unanimity or affine operation [23]. This completes the proof of Theorem 47. ■

8.4.3 Homogeneous algebras

We have studied homogeneous algebras in Section 7.3. Homogeneous algebras have a natural interpretation in terms of quantified formulas, they correspond to formulas over basis containing all the permutation relations. Recall that a permutation in D is any relation of the form $\pi^\square$ for some permutation in D.

For basis with permutation relations the analysis is identical to Theorem 35. The next theorem summarizes our results.

Theorem 49 *Let Γ be a finite set of relations over D. If* $\mathrm{Pol}(\Gamma)$ *contains:* (1) *a near-unanimity operation,* (2) *a coset generating operation, or*

(3) *operations p, and g, where $(D; p, G)$ is a* **MPP**, *then* $\exists\forall\text{-}\,\mathrm{Form}_C(\Gamma \cup \{\pi\square : \pi$ *bijection in* $D\})$ *is polynomially learnable, otherwise* $\exists\text{-}\,\mathrm{Rel}(\Gamma \cup \{\pi\square : \pi$ *bijection in* $D\})$ *is not polynomially predictable with membership queries under cryptographic assumptions. More precisely, the class* $\exists\forall\text{-}\,\mathrm{Form}_C(\Gamma \cup \{\pi\square : \pi$ *bijection in* $D\})$ *is polynomially learnable if and only if*

- $\{s, d\} \cap \mathrm{Pol}(\Gamma) \neq \emptyset$ *if* $|D| \notin \{3, 4\}$.
- $\{s, d, x + y + z\} \cap \mathrm{Pol}(\Gamma) \neq \emptyset$, *where* $(D, +)$ *is a group of exponent* 2, *if* $|D| = 4$.
- $\{s, d, r_3\} \cap \mathrm{Pol}(\Gamma) \neq \emptyset$, *if* $|D| = 3$.

Chapter 9

Some Applications of Post's Classification

9.1 Introduction

In previous chapters we have shown how some concepts in clone theory can be applied to computational problems such as learnability of generalized quantified boolean formulas or constraint satisfaction problems. Unfortunately, at the present moment, the knowledge of the clone structure for domains with more than two elements is still incomplete. Nevertheless, the clone structure for the binary domain is completely known and was described by Post [76]. We can use this knowledge to study and solve completely computational problems over the boolean domain. In this chapter we show some applications of Post's classification.

First, we use Post's classification of boolean clones to provide an alternative and simpler proof of (part of) the dichotomy results in Chapters 3 and 4. Recall that the most difficult part in the proof of the dichotomy results presented in Chapters 3 and 4 concerns merely the expressivity power of some sets of formulas and is proven by an exhaustive and involved case analysis. In Sections 9.2 and 9.3 we simplify these proofs by using Post's classification. The underlying intuition is that the expressivity power of the set of generalized formulas over a basis is, under some technical assumption, equivalent to the co-clone generated by the basis, and in consequence, Post's description contains a detailed analysis of all possible cases. Thus, we save time by using Post's classification instead of redoing the case analysis again and again.

This approach was used in [48] to give an alternative proof of Shaeffer's classification.

Furthermore we study two new computational problems over boolean domains. The inverse satisfiability problems studied in Section 9.4 consists in deciding whether a given relation can be obtained as the set of solutions of a generalized formula over a basis. In the constraint satisfaction setting this problem has been called *structure identification* [27]. We study the problem over any domain, obtaining some general conditions that guarantee the tractability of the problem. Finally, we obtain particular results in the boolean case as an application of a particular feature of the boolean clones: the fact that any boolean clone is finitely generated.

Finally, in Section 9.5 we study the learnability of formulas obtained by restricting the set of possible basic functions that can be used to build them. This class of formulas differs from generalized quantified formulas with the fact that the functions in the basis are combined using composition instead of quantification and conjunction, i.e., now formulas are trees. We show that this class of formulas is related with clones (recall that the class of generalized formulas is related with co-clones) and therefore we can use Post's classification in order to obtain a complete classification of their learnability.

9.2 Learnability of Quantified Boolean Formulas Revisited

In this section we review some results about the learnability of quantified boolean formulas presented in Chapter 3 under the perspective provided by its connection with clone theory, in order to simplify the proof .

Recall (Theorem 4) that for every finite set Γ of boolean relations, the class of formulas $\exists\forall$- $\mathrm{Form}_C(\Gamma)$ is polynomially learnable with equivalence queries if Γ is bijunctive, weakly monotone (WM), weakly antimonotone (WA) or affine. Otherwise, $\exists\forall$- $\mathrm{Form}_C(\Gamma)$ is not polynomially predictable even with membership queries under cryptographic assumptions.

The proof of this result (Theorem 4) shown in Chapter 3 can be splitted in three parts: **(1)** If a basis Γ is bijunctive, weakly monotone, weakly antimonotone or affine, then $\exists\forall$- $\mathrm{Form}_C(\Gamma)$ is polynomially learnable. **(2)** If, Γ can "simulate" $[x \vee y \vee \overline{z}]$ or $[\overline{x} \vee \overline{y} \vee z]$ then $\exists\forall$- $\mathrm{Form}_C(\Gamma)$ is not polynomially predictable with membership queries. Finally, **(3)** For every basis Γ, one of conditions (1) or (2) holds.

Parts (1) and (2) are concerned only with the complexity aspects of the problem. Part (3) relates uniquely the expressivity power of the class of quantified formulas over a basis.

In this section we show how clone theory and, in particular, Post's classification can be used to prove Theorem 4. First, the learnability of bijunctive, weakly monotone and weakly monotone and affine basis can be explained as a corollary of a more general result concerning the learnability of subuniverses of power algebras for certain classes of algebras, proved in Chapter 8, obtaining an alternative proof of Part (1). Secondly, part (3) can be proven by using Post's classification avoiding the involved case analysis presented in the original proof in Chapter 3.

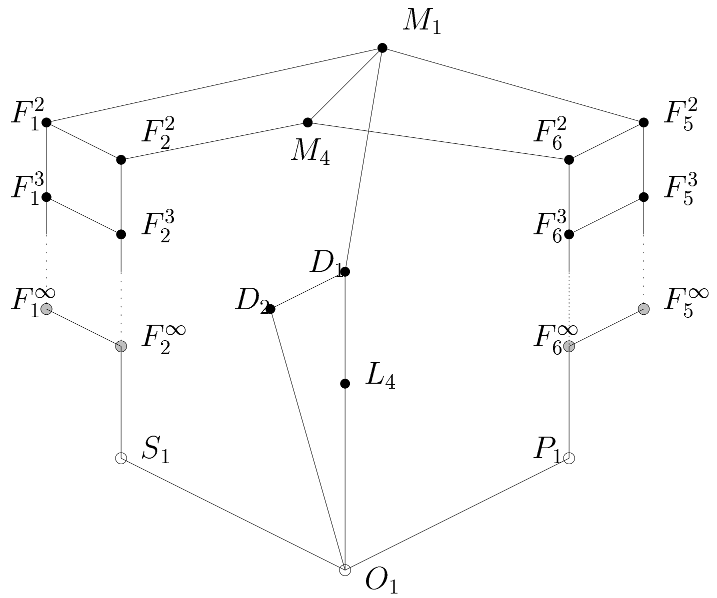

Figure 9.1: Post Lattice restricted to idempotent operations

Let us start by introducing the elements of Post's classification that we will use in our analysis. Diagram 9.1 represents the idempotent Post lattice, that is, the lattice of clones on a 2-element set (Figure 5.1) with idempotent operations only. We consider only idempotent operations because, as a consequence of Theorem 44, the learnability of generalized quantified formulas over a basis S depends only on the idempotent polymorphims of S, when the use of constants is allowed.

Nodes in figure 9.1 are labeled according to Post's notation (see Chapter 5 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. Actually, every learnable class in the boolean case is associated to a near-unanimity or affine operation. The following table establishes the correspondence between these clones and the learnable classes in the boolean case, which can be proved directly from the definitions.

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	$\varphi_{F_6^k}(x_1, \ldots, x_{k+1}) = \bigvee_{j=1}^{k+1}(x_1 \wedge .. \wedge x_{j-1} \wedge x_{j+1} \wedge .. \wedge x_{k+1})$
affine	$\varphi_{L_4}(x, y, z) = x \oplus y \oplus z$

Every entry in the table, constituted by a class of basis C and an operation φ, has to be interpreted in the following way: a basis Γ belongs to the class C iff $\varphi \in \mathrm{Pol}(\Gamma)$.

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 these classes follows directly from Theorems 38 and 39. As the lattice is ordered from bottom to top according to inclusion, the remainder of the black nodes contain also a near-unanimity or an affine operation inherited from some ancestor.

The remainder of the clones are depicted as white nodes in figure 9.1. 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 $[x \vee y \vee \overline{z}]$ and, by Corollary 3, the associated set of quantified formulas is not polynomially predictable with membership queries under cryptographic assumptions. This simple reasoning gives us an alternative proof of the involved case analysis in Chapter 3.

9.3 Satisfiability of Quantified Boolean Formulas

In Chapter 4 we proved a dichotomy theorem (Theorem 12) for the satisfiability of constant-free quantified boolean formulas. The core of the proof is a result stated in Theorem 11 concerning only the expressivity power of quantified formulas. In this section we give give an alternative proof based on Post's classification of Theorem 11. First, recall the content of Theorem 11:

Let Γ be a finite basis of boolean relations. Than at least one of the following conditions holds:

a.- Every relation in Γ is 0-affine.

b.- Every relation in Γ is 1-affine.

c.- Every relation in Γ is equality antimonotone.

d.- Every relation in Γ is equality monotone.

e.- The logical relations $[x]$ and $[\overline{x}]$ are contained in $\exists\forall\text{-}\operatorname{Rel}(\Gamma)$.

f.- The logical relation $[x \not\equiv y]$ is contained in $\exists\forall\text{-}\operatorname{Rel}(\Gamma)$ and every relation in Γ is complementive.

Furthermore, if Γ satisfies at least one of conditions (a), (b), (c), or (d), then, Γ satisfies none of conditions (e) or (f), and conversely, if Γ satisfies at least one of conditions (e) or (f), then Γ satisfies none of conditions (a), (b), (c), or (d).

The difficult part of the proof consists in showing that if none of conditions $(a) - (d)$ is satisfied, then condition (e) or (f) holds. In this section we give an alternative proof of the involved case analysis given in Chapter 4 based on Post's classification. First, we characterize the families of relations occurring in the theorem in terms of polymorphisms.

Lemma 39 *Let Γ be a set of boolean relation. Then:*

a.- Γ *is* 0*-affine iff* $L_3 \subseteq \operatorname{Pol}(\Gamma)$*.*

b.- Γ *is* 1*-affine iff* $L_2 \subseteq \operatorname{Pol}(\Gamma)$*.*

c.- Γ *is equally antimonotone iff* $F_8^\infty \subseteq \operatorname{Pol}(\Gamma)$*.*

c.- Γ is equally monotone iff $F_4^\infty \subseteq \mathrm{Pol}(\Gamma)$.

Proof. Direct from the definitions. We show an overview of the proof of item (a). Proof of items $(b)-(d)$ are analogous. Let R be a 0-affine relation. Thus R can be expressed as the set of solutions of a system of equations over the two element field $\{0,1\}$. Since the set of solutions of every individual equation is preserved by $\varphi_{L_4}(x,y,z) = x \oplus y \oplus z$, the set of solutions of a system is also preserved by φ_{L_4}. Furthermore, since the system is homogeneous, if it has a solution, then it has the solution identically zero. This fact is equivalent to being preserved by operation $\varphi_{O_6} = 0$. Finally, notice that φ_{L_4} and φ_{O_6} generate L_3. This completes the proof in one direction.

Conversely, if a relation R is preserved by all the operations in L_3, then R can be obtained with a sequence of projections, selections and cartesian products using $R_{L_1} = \{(x,y,z,t) = x \oplus y \oplus z \oplus t = 0\}$, and $R_{C_3} = \{\langle 0\rangle\}$ as elementary relations. Clearly, the previous condition implies the existence of a system of equations equivalent to R. ■

Now we study which portion of Post's lattice is covered by 0-affine, 1-affine, equality monotone and equality antimonotone bases. With a simple inspection of Post's lattice we can characterize the clones not containing any of the following clones: L_3, L_2, F_4^∞, F_8^∞. More precisely, any clone not containing any of the previous clones is contained in one of the following clones: C_4, M_1, D_3 and O_9. Now, we can change the view and look at the co-clones. Let us do a case analysis:

- Let Γ be a finite set of boolean relations such that $\mathrm{Pol}(\Gamma) \subseteq C_4$, then $R_{C_2} = \{\langle 1\rangle\} = [x]$ and $R_{C_3} = \{\langle 0\rangle\} = [\overline{x}]$ are in $\langle\Gamma\rangle_{re}$. In consequence, R_{C_2} and R_{C_3} are in $\exists\text{-}\mathrm{Rel}(\Gamma)$ and we are done.

- By a similar reasoning, we can infer that if $\mathrm{Pol}(\Gamma) \subseteq M_1$, then $R_{M_1} = \{\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,1\rangle\} = [\overline{x} \vee y]$ is in $\exists\text{-}\mathrm{Rel}(\Gamma)$, and therefore we have $[y] = [\forall x(\overline{x} \vee y)]$ and $[\overline{x}] = [\forall y(\overline{x} \vee y)]$ are in $\exists\forall\text{-}\mathrm{Rel}(\Gamma)$.

- If $\mathrm{Pol}(\Gamma) \subseteq D_3$, then $R_{D_3} = \{\langle 0,1\rangle, \langle 1,0\rangle\} = [x \neq y]$ is in $\exists\text{-}\mathrm{Rel}(\Gamma)$. At this point we should distinguish between two sets of clones. If $\mathrm{Pol}(\Gamma) \subseteq C_4$, that is $\mathrm{Pol}(\Gamma) \in \{D_1, D_2, O_1\}$ then we are done. Otherwise, $\mathrm{Pol}(\Gamma) \in \{D_3, L_5, O_4\}$. Notice that in all cases $\varphi_{O_4}(x) = \overline{x}$ is in $\mathrm{Pol}(\Gamma)$. Finally, notice that if φ_{O_4} preserves Γ iff Γ is complementive.

- Finally, if $\mathrm{Pol}(\Gamma)$ is in O_9, then $R_{O_9} = \{|t|_1 \mathrm{mod} 4 = 0\}$ is in $\exists\text{-}\mathrm{Rel}(\Gamma)$. In consequence $[x \neq y] = [\forall z \exists t R(x,y,z,z,z,t,t,t)]$ is in $\exists\forall\text{-}\mathrm{Rel}(\Gamma)$. Furthermore, we have that $\mathrm{Pol}(\Gamma) \in \{O_1, O_5, O_6, O_4, O_8, O_9\}$. If $\mathrm{Pol}(\Gamma) \in \{O_4, O_8, O_9\}$, then $\varphi_{O_4}(x) = \overline{x}$ preserves Γ and, in consequence, Γ is complementive. In the other cases $\mathrm{Pol}(\Gamma) \subseteq M_1$ and we are done.

9.4 Inverse Satisfiability Problem

9.4.1 Introduction

From a general point of view, *structure identification* is the problem of finding a computation description of some relational data. To our knowledge, there has been very few attempts to study this problems from a formal point of view. In [27], Dechter and Pearl presented a general framework for the problem and some interesting relationships with the learning problem. The main problem addressed in [27] is the following: given a relation R representing a set of satisfying assignments, find a constraint network where the solutions are precisely the tuples in R.

In this section we address a similar problem, called *inverse satisfiability problem*, consisting in finding a generalized formula over a basis that satisfies a given set of assignments, when both the basis and the set of assignments can be part of the input. Notice that for a fixed basis and quantifier-free formulas, the problem can be formulated in the following way: given a set of solutions R, decide whether or not there exists a constraint network $\mathcal{P}$ where every constraint relation belongs to a fixed basis and such that $\mathcal{P}$ expresses R. This problem was solved completely in the boolean case by Kavvadias and Sideri [54] under the assumption that constants are available without using the connection with clone theory. In this section, we study *inverse satisfiability problems* taking advantage of its connection with clone theory. We give results for the non-uniform problems obtained fixing either the relation or the basis. Finally, we obtain an extension of the results in [54] to quantified formulas as an application of Post's classification.

9.4.2 Definitions and Preliminaries

Definition 26 Let $\Delta \subseteq \mathcal{R}^A$ be a set of finite relations over a finite domain A and let $\mathcal{S} \subseteq \mathcal{P}(\mathcal{R}^A)$ be a class of finite sets of relations over A. The *inverse satisfiability problem for generalized quantified formulas with constants*, $\exists\forall$- $\mathrm{InvSAT}_C(\Delta, \mathcal{S})$ is the class of decision problems with:

- Instance: A relation $R \in \Delta$ and a finite basis $\Gamma \in \mathcal{S}$.
- Question: There exists a formula $\Phi \in \exists\forall\text{-}\operatorname{Form}_C(\Gamma)$ such that $R = [\Phi]$?.

If in the previous definition we replace "$\exists\forall\text{-}\operatorname{Form}_C(\Gamma)$" by another class of formulas, such as "$\exists\forall\text{-}\operatorname{Form}(\Gamma)$", we obtain the inverse satisfiability problem for constant-free generalized quantified formulas, denoted by $\exists\forall\text{-}\operatorname{InvSAT}(\Delta, \mathcal{S})$. Analogous definitions can be given for $\exists\text{-}\operatorname{InvSAT}_C(\Delta, \mathcal{S})$, $\exists\text{-}\operatorname{InvSAT}(\Delta, \mathcal{S})$, $\operatorname{InvSAT}_C(\Delta, \mathcal{S})$, and $\operatorname{InvSAT}(\Delta, \mathcal{S})$.

Fixing some of the components of the input we obtain non-uniform problems. First, for a fixed finite basis Γ of relations over A, $\exists\forall\text{-}\operatorname{InvSAT}_C(\mathcal{R}^A, \{\Gamma\})$ (we will use also $\exists\forall\text{-}\operatorname{InvSAT}_C(\mathcal{R}^A, \Gamma)$) is the problem of deciding whether a given relation $R \in \mathcal{R}^A$ (the input) belongs to $\exists\forall\text{-}\operatorname{Form}_C(\Gamma)$. Equivalently, for a fixed relation $R \in \mathcal{R}^A$, $\exists\forall\text{-}\operatorname{InvSAT}_C(\{R\}, \mathcal{P}(\mathcal{R}^A))$ (we will use also $\exists\forall\text{-}\operatorname{InvSAT}_C(R, \mathcal{P}(\mathcal{R}^A))$)is the problem of deciding whether a given basis Γ (the input) can express R.

The underlying idea of our approach is the following: the problem of deciding the membership of a relation R to the class $[\mathcal{F}]$ where $\mathcal{F}$ is a class of formulas over a basis A resembles the problem of deciding the membership of a relation to a restricted co-clone (content of Lemma 40). More precisely, for every class of formulas closed under renaming of variables, conjunction and existential quantification, its associated class of relations is a restricted co-clone. Recall that a restricted co-clone on A is a class of relations that contains the relation A, and closed under cartesian product and restricted projection (see Section 5.4.2).

Thus, we will see that the problem of deciding whether a relation R belongs to a restricted co-clone Ψ can be reduced, under some technical assumptions, to the problem of deciding whether a *canonical* relation R' belongs to Ψ (content of Lemma 41). This latter question can be solved (Theorem 15) by testing whether every operation in $\operatorname{Pol}(\Psi)$ preserves R' (i.e, $\operatorname{Pol}(\Psi) \subseteq \operatorname{Pol}(R')$).

Let us start by showing that some classes of relations defined in Section 2.1.1 are actually restricted co-clones. First, we introduce some notation.

For any n-ary relation R, any index $1 \leq i \leq n$ and any value $a \in A$ the *constantification* $R_{a \leftarrow i}$ is given by:

$$R_{a \leftarrow i} = \{\langle x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n \rangle : \langle x_1, \ldots, x_{i-1}, a, x_{i+1}, \ldots, x_n \rangle \in R\}.$$

Given a set of relations Γ, we denote $\langle\Gamma\rangle_c$ the minimum set containing Γ and closed under constantification.

For any n-ary relation R and any index $1 \leq i \leq n$ the *universal projection* $\operatorname{upr}_i R$ is given by:

$$\operatorname{upr}_i R = \bigcap_{a \in A} R_{a \to i}$$

Given a set of relations Γ, we denote $\langle\Gamma\rangle_u$ the minimum set containing Γ and closed under universal projection. Notice that if Γ is finite, then $\langle\Gamma\rangle_c$ and $\langle\Gamma\rangle_u$ are finite as well.

The following lemma shows that for every class of formulas defined in Section 2.1.1 closed under universal quantification, its associated class of formulas is a restricted co-clone.

Lemma 40 *For every class Γ of relations over A we have*

- $\exists\text{-}\operatorname{Rel}(\Gamma) = \langle\Gamma\rangle_{re}$,
- $\exists\forall\text{-}\operatorname{Rel}_C(\Gamma) = \exists\text{-}\operatorname{Rel}_C(\Gamma) = \langle\Gamma_1\rangle_{re}$, *where* $\Gamma_1 = \langle\Gamma\rangle_c$
- $\exists\forall\text{-}\operatorname{Rel}(\Gamma) = \langle\Gamma_2\rangle_{re}$, *where* $\Gamma_2 = \langle\Gamma\rangle_u$

Proof. Proof easy and omitted. ■

Once we have related some classes of formulas to restricted co-clones, now we introduce the main result (Lemma 41) we will use to study membership to a restricted co-clone. First we introduce some terminology.

Recall that a k-ary relation R is canonical if and only if for all $1 \leq i \neq j \leq k$ there exists some tuple $t \in R$, such that $t[i] \neq t[j]$. Given any arbitrary k-ary relation R, consider any arbitrary list of indices $i_1, \ldots, i_j \;\; j \leq k$ containing exactly one of the representatives of the equivalence class given by the following equivalence relation: $i \equiv j \;\; (1 \leq i, j \leq k)$ iff for all $t \in R$ $t[i] = t[j]$. Clearly, $\operatorname{pr}_{i_1,\ldots,i_j} R$ is canonical. We call $\operatorname{pr}_{i_1,\ldots,i_j} R$ a *canonical relation associated to R*. Every binary relation $\operatorname{pr}_{j,l} R \;\; j \neq l, j \equiv l$ is referred to as a *subidentity relation associated to R*.

Lemma 41 *Let R be a relation over A and let Γ be a set of relations over D. Then* (1) *R belongs to the restricted co-clone generated by Γ if and only if* (2) *every canonical relation associated to R and every subidentity relation associated to R belongs to the restricted co-clone generated by Γ.*

Proof. Proof easy and omitted. ∎

Recall that from the last remark in the proof of Theorem 15 that a canonical relation R belongs to the restricted co-clone generated by a set of relations Γ if and only if $\mathrm{Pol}(\Gamma) \subseteq \mathrm{Pol}(R)$.

We finish this section by giving some results about the complexity of the general inverse satisfiability problem when no restrictions are placed. Actually very little is known. We show that the problem is decidable. We prove it for quantified-free formulas and for quantified formulas separately.

First, we show that problem $\mathrm{InvSAT}(\mathcal{R}^A, \mathcal{P}(\mathcal{R}^A))$ is decidable (the case $\mathrm{InvSAT}_C(\mathcal{R}^A, \mathcal{P}(\mathcal{R}^A))$ is solved analogously). To prove it, notice that given a k-ary relation R it is possible to calculate the best candidate Φ in $\mathrm{Form}(\Gamma)$ to satisfy $[\Phi] = R$. The formula Φ is given by the conjunction of all the expressions of the form $S(x_{i_1}, \ldots, x_{i_l})$ where S is a l-ary relation in Γ, $x_{i_1}, \ldots, x_{i_l}$ are variables in $\{x_1, \ldots, x_k\}$ such that for every $\langle a_1, \ldots, a_k \rangle \in R$, $\langle a_{i_1}, \ldots, a_{i_l} \rangle \in S$. By construction, $R \subseteq [\Phi]$. Finally we check whether $R = [\Phi]$ by testing all the k-ary assignments.

We can extend the previous construction to quantified classes of formulas by bounding the number of quantified variables. We will prove it for $\exists\text{-}\mathrm{InvSAT}(\mathcal{R}^A, \mathcal{P}(\mathcal{R}^A))$. As a consequence of Lemma 40, the same result holds for all the quantified classes of formulas.

We bound the number of quantified variables using a simple and nice technique that appears in [73]. Let R be a k-ary relation and let $\Phi = \exists y_1, \ldots, y_m \Psi(x_1, \ldots, x_k, y_1, \ldots, y_m)$ be a formula in $\exists\text{-}\mathrm{Form}(\Gamma)$ with k free variables $x_1, \ldots, x_k$ and m existentially quantified variables $y_1, \ldots, y_m$ such that m is minimal and $R = [\Phi]$. Associate to every satisfying assignment of the k free variables an assignment of the m bounded variables that actually satisfies the quantifier-free formula Ψ. Thus, we have a function $f : R \to A^m$ expressing the previous correspondence. If for two different quantified variables y_i, y_j $(1 \leq i \neq j \leq m)$, the corresponding components in f are identical (i.e. $f_i(t) = f_j(t)$ for all $t \in R$), then the formula obtained identifying y_i and y_j is equivalent to $[\Phi]$. Thus, the total number of quantified variables is bounded by the total amount of functions $f : R \to A$. In consequence we get the bound $m \leq |A|^{|R|} \leq |A|^{|A|^k}$. Summarizing

Lemma 42 *For every finite domain A, the following problems are decidable*

- $\mathrm{InvSAT}(\mathcal{R}^A, \mathcal{P}(\mathcal{R}^A))$,
- $\mathrm{InvSAT}_C(\mathcal{R}^A, \mathcal{P}(\mathcal{R}^A))$,

- $\exists$- $\mathrm{InvSAT}(\mathcal{R}^A, \mathcal{P}(\mathcal{R}^A))$,
- $\exists$- $\mathrm{InvSAT}_C(\mathcal{R}^A, \mathcal{P}(\mathcal{R}^A))$,
- $\exists\forall$- $\mathrm{InvSAT}(\mathcal{R}^A, \mathcal{P}(\mathcal{R}^A))$,
- $\exists\forall$- $\mathrm{InvSAT}_C(\mathcal{R}^A, \mathcal{P}(\mathcal{R}^A))$.

Although we know that the inverse satisfiability problem is decidable, the bound given by the previous algorithms (specially in the quantified case) is quite large and it is possible that it could be reduced using more sophisticated algorithms.

On the other hand, we know very little about lower bounds for the inverse satisfiability problem. It is known that $\mathrm{InvSAT}_C(\mathcal{R}^{\mathcal{B}}, \{\Gamma\})$ is co-NP-hard when Γ is a finite boolean basis which does not fall in any of the following classes: horn, antihorn, affine, and bijunctive [54].

It is an interesting question to reduce the big gap between lower and upper bounds for the inverse satisfiability problem.

9.4.3 The Complexity of Inverse Satisfiability Problems for Fixed Bases

Let us study the complexity of the non-uniform inverse satisfiability problem for quantified formulas obtained when the basis Γ is fixed. Let A be the universe of the relations in R. The motivation is a previous result of the complexity of $\mathrm{InvSAT}_C(\mathcal{R}^{\mathcal{B}}, \Gamma)$ for boolean bases Γ due to Kavvadias and Sideri [54]. More precisely, in [54] it is show that $\mathrm{InvSAT}_C(\mathcal{R}^{\mathcal{B}}, \Gamma)$ is polynomially decidable iff Γ is horn, antihorn, bijunctive or affine.

In this section we study the inverse satisfiability problem for quantified formulas. We show that for *all* boolean basis, the problem is polynomial-time decidable. We prove a more general result.

We say that a clone C is *finitely generated* if there exists some finite set of functions $f_1, \dots, f_k$ such that $C = \langle f_1, \dots, f_k \rangle_o$.

Recall that, by virtue of Lemma 40, for some of the classes of formulas defined in Section 2.1.1 such as $\exists$- $\mathrm{Form}(\cdot)$, $\exists$- $\mathrm{Form}_C(\cdot)$, $\exists\forall$- $\mathrm{Form}(\cdot)$, and $\exists\forall$- $\mathrm{Form}_C(\cdot)$, their associated class of relations is a restricted co-clone. In consequence we focus on the study of the problem of deciding the membership to the restricted co-clone generated by a set of relations.

First, notice that for a fixed domain we have a finite number of binary relations R with $R \subseteq \mathrm{eq}_A$. Thus, the membership of every such relation R in a fixed restricted co-clone $\langle\Gamma\rangle_{re}$ can be precomputed. Thus, we will care only about canonical relations.

For canonical relations, membership to a restricted co-clone is determined completely by the set of polymorphisms. That is, given a class $\mathcal{F}$ of formulas such that $[\mathcal{F}] = \langle\Delta\rangle_{re}$, we have $R \in [\mathcal{F}]$ iff $\mathrm{Pol}(\Delta) \subseteq \mathrm{Pol}(R)$ whenever R is canonical.

Furthermore, if $\mathrm{Pol}(\Delta)$ is finitely generated, then we have a polynomial-time algorithm to decide the membership of a canonical relation to $\langle\Delta\rangle_{re}$: Let $\{f_1, \ldots, f_m\}$ be a set of functions that generates $\mathrm{Pol}(\Gamma)$, i.e., $\langle f_1, \ldots, f_m\rangle_o = \mathrm{Pol}(\Delta)$. Notice that for a fixed function f of arity k, decide whether $f \in \mathrm{Pol}(R)$ can be done in polynomial time by cheking if $f(t_1, \ldots, t_k)$ for all $t_1, \ldots, t_k$ (not neccesarily different) tuples in R.

Thus, we get the following result

Theorem 50 *Let Γ be a (fixed) restricted co-clone on A, such that $\mathrm{Pol}(\Gamma)$ is finitely generated. Then the problem of deciding the membership of a relation R (the input) to Γ is decidable in polynomial-time.*

In particular,

Corollary 8 *Let Γ be a (non necessarily finite) set of relations over A, such that $\mathrm{Pol}(\Gamma)$ is finitely generated. Then $\exists\text{-}\mathrm{InvSAT}(\mathcal{R}^A, \Gamma)$ is polynomially decidable.*

Corollary 9 *Let Γ be a (non necessarily finite) set of relations over A, such that clone $\mathrm{Pol}(\langle\Gamma\rangle_c)$ is finitely generated. Then $\exists\text{-}\mathrm{InvSAT}_C(\mathcal{R}^A, \Gamma)$, $\exists\text{-}\mathrm{InvSAT}_C(\mathcal{R}^A, \Gamma)$, and $\exists\forall\text{-}\mathrm{InvSAT}_C(\mathcal{R}^A, \Gamma)$ are polynomially decidable.*

Corollary 10 *Let Γ be a (non necessarily finite) set of relations over A, such that $\mathrm{Pol}(\langle\Gamma\rangle_u)$ is finitely generated. Then $\exists\forall\text{-}\mathrm{InvSAT}(\mathcal{R}^A, \Gamma)$ is polynomially decidable.*

From Post's classification we have that every clone in the boolean domain is finitely generated. Then

Corollary 11 *Let Γ be a (non necessarily finite) set of boolean relations, then the following problems are polynomial-time decidable:*

- $\exists\forall$- $\text{InvSAT}_C(\mathcal{R}^{\mathcal{B}}, \Gamma)$,
- $\exists\forall$- $\text{InvSAT}(\mathcal{R}^{\mathcal{B}}, \Gamma)$,
- $\exists$- $\text{InvSAT}_C(\mathcal{R}^{\mathcal{B}}, \Gamma)$, *and*
- $\exists$- $\text{InvSAT}(\mathcal{R}^{\mathcal{B}}, \Gamma)$,

Thus, the inverse satisfiability problem for quantified boolean formulas is polynomial-time decidable. The only remaining case for boolean formulas is to classify completely the computational complexity of $\text{InvSAT}(\mathcal{R}^{\mathcal{B}}, \Gamma)$, still open.

9.4.4 The Complexity of Inverse Satisfiability Problems for a Fixed Relation

Let us study now the non-uniform problem obtained fixing the relation instead of the basis. We will show that the problem can be reduced to some simpler problems.

First, notice (Lemma 40) that for every k-ary relation R and every class $\mathcal{F}$ of constant-free existentially quantified formulas, if $R \in [\mathcal{F}]$, then there exists some formula $\Phi \in \mathcal{F}$ such that $[\Phi] = R$ and Φ has at most $m = |A|^{|R|}$ quantified variables. Furthermore, for a fixed R, the total number of $(k+m)$-ary relations that produce R projecting, say, over $1, \ldots, k$ is finite (although it can be very large). Thus, $R \in [\mathcal{F}]$ iff some of such relations can be generated by a quantifier-free formula in $\mathcal{F}$.

Thus,

Lemma 43 *Let $R \in \mathcal{R}^A$ be a relation over A. If* $\text{InvSAT}(R, \mathcal{P}(\mathcal{R}^A))$ *is polynomially solvable, then* $\exists$- $\text{InvSAT}(R, \mathcal{P}(\mathcal{R}^A))$ *is polynomially solvable.*

Furthermore, we will show that if it is possible to solve in polynomial time the problem for every relation $R \subseteq eq_A$, then the general problem for every $R \in \mathcal{R}^A$ is polynomially solvable. Let us start with canonical relations.

Theorem 51 *For any fixed canonical relation R, the problem of deciding whether R belongs to $\langle\Delta\rangle_{re}$ for a given Δ is polynomial-time decidable*

Proof. The problem is equivalent to deciding whether $\text{Pol}(\Delta) \subseteq \text{Pol}(R)$. We make use of a previous result in universal algebra [37]. First we introduce some notation.

Let H a clone on A. A set of relations $\Gamma \subseteq \mathcal{R}^A$ is called H-polyrigid if $\mathrm{Pol}(\Gamma) \subseteq H$. In consequence, $R \in \langle\Gamma\rangle_{re}$ iff Γ is $\mathrm{Pol}(R)$-polyrigid.

A clone $F \subseteq \mathcal{F}^A$ is called H-minimal if F is not contained in H and does not contain properly another clone $F' \not\subseteq H$. From [37] we have,

Theorem 52 *Let $H = \mathrm{Pol}(R)$ for some finite relation $R \in \mathcal{R}^A$. Then a set of relations Γ is H-polyrigid iff every H-minimal clone F and an (arbitrary chosen) $f \in F \setminus H$ there exists a relation $S \in \Gamma$ such that f does not preserve S. Furthermore, the set of H-minimal clones is finite.*

Thus, let R be a fixed relation and let $H = \mathrm{Pol}(R)$, its clone of polymorphisms. Since the set of H-minimal clones is finite, we can fix an operation $f_F \in F \setminus H$ for every H-minimal clone F.

The algorithm tests whether this set of operations preserves Δ. That is $R \in \langle\Delta\rangle_{re}$ iff $f_F \in \mathrm{Pol}(\Gamma)$ for every H-minimal clone F. ■

The previous result gives a polynomial-time algorithm for some classes of formulas $\mathcal{F}$ with $[\mathcal{F}] = \langle\Delta\rangle_r$, provided it is possible to decide efficiently whether a fixed operation preserves Δ. The previous condition is satisfied by $\exists$- $\mathrm{Form}(\Gamma)$ for every finite Γ, since $\exists$- $\mathrm{Form}(\Gamma) = \langle\Gamma\rangle_{re}$. It is also satisfied by $\exists\forall$- $\mathrm{Form}_C(\Gamma)$ since $\mathrm{Pol}(\exists\forall\text{-}\,\mathrm{Rel}_C(\Gamma)) = \mathrm{Idem}(\mathrm{Pol}(\Gamma))$.

Corollary 12 *Let R be a canonical relation over A. Thus the following problems are polynomial-time decidable:*

- $\exists\forall$- $\mathrm{InvSAT}_C(R, \mathcal{P}(\mathcal{R}^A))$
- $\exists$- $\mathrm{InvSAT}_C(R, \mathcal{P}(\mathcal{R}^A))$
- $\exists$- $\mathrm{InvSAT}(R, \mathcal{P}(\mathcal{R}^A))$

Then, by virtue of Lemma 41, we have

Theorem 53 *Let $R \in \mathcal{R}^A$ be a relation on A such that for every subequality S associated to R, $\exists\forall$- $\mathrm{InvSAT}_C(S, \mathcal{P}(\mathcal{R}^A))$ (resp. $\exists$- $\mathrm{InvSAT}_C(S, \mathcal{P}(\mathcal{R}^A))$, $\exists$- $\mathrm{InvSAT}(S, \mathcal{P}(\mathcal{R}^A))$) can be decided in polynomial time, then the problem $\exists\forall$- $\mathrm{InvSAT}_C(R, \mathcal{P}(\mathcal{R}^A))$ ($\exists$- $\mathrm{InvSAT}_C(R, \mathcal{P}(\mathcal{R}^A))$, $\exists$- $\mathrm{InvSAT}(R, \mathcal{P}(\mathcal{R}^A))$) is polynomial-time decidable.*

Corollary 13 *If for every relation $R \subseteq \mathrm{eq}_A$, $\exists\forall$- $\mathrm{InvSAT}_C(R, \mathcal{P}(\mathcal{R}^A))$ (resp. $\exists$- $\mathrm{InvSAT}_C(S, \mathcal{P}(\mathcal{R}^A))$, $\exists$- $\mathrm{InvSAT}(S, \mathcal{P}(\mathcal{R}^A))$) is polynomial-time decidable, then for each $R \in \mathcal{R}^A$, $\exists\forall$- $\mathrm{InvSAT}_C(R, \mathcal{P}(\mathcal{R}^A))$ (resp. $\exists$- $\mathrm{InvSAT}_C(R, \mathcal{P}(\mathcal{R}^A))$, $\exists$- $\mathrm{InvSAT}(R, \mathcal{P}(\mathcal{R}^A))$) is polynomial-time decidable.*

9.5 Learnability of Boolean Formulas and Circuits

9.5.1 Preliminaries

In this section we study also the learning complexity of some classes of boolean formulas obtained placing also some restrictions in the elementary boolean functions that can be used to build the formulas.

More precisely, let $F = \{f_1, \ldots, f_m\}$ be a finite set of boolean functions. A formula in $\mathrm{FOR}(F)$ can be any of (a) a boolean variable, or (b) an expression of the form $f(g_1, \ldots, g_k)$ where f is a k-ary function in F and $g_1, \ldots, g_k$ are formulas in $\mathrm{FOR}(F)$.

In general, boolean formulas are constructed by using elementary functions from a complete basis, generally $\{\mathrm{AND}, \mathrm{OR}, \mathrm{NOT}\}$. In this section we will allow formulas to use as a basis any arbitrary set of boolean functions. For example, consider the problem of learning a monotone boolean formula. Every such formula can be expressed as a formula in the class $\mathrm{FOR}(\{\mathrm{AND}, \mathrm{OR}\})$.

This class of formulas is similar to the class of generalized quantified formulas in the sense that every basis or finite set of functions defines a family of formulas. Notice that, in the boolean case, considering functions or relations is irrelevant, since every boolean function defines a boolean relation and vice versa. The main difference relies in the way in which this basic functions are combined. In generalized quantified formulas ($\exists\forall$- $\mathrm{Form}(F)$), these basic functions are combined basically with conjunctions and quantification whereas in this new class of formulas ($\mathrm{FOR}(F)$) these basic functions are combined through composition. As a consequence of this fact the mathematical structures involved in the study of the expressivity power of these classes of formulas is different. The class of generalized formulas is associated to co-clones and the class studied in this section is associated to clones. This fact justifies the choice of relations or functions in each case.

The main result of this chapter characterizes the complexity of learning $\mathrm{FOR}(F)$ for every finite set F of boolean functions. The most striking feature of this characterization is that for any F, $\mathrm{FOR}(F)$ is either polynomially learnable with equivalence or membership queries alone or, under some cryptographic assumptions, not polynomially predictable even with membership queries.

Furthermore, we give an interesting classification of the polynomially learnable classes. We show that, in a sense that will be made precise later, FOR(F) is polynomially learnable if and only if functions in F are all disjunctions of variables or all are conjunctions of variables, or all are linear combinations of variables.

There is another rather special feature of this result. Learnability of boolean formulas over a basis F depends only on the set of functions that can be expressed as a formula in FOR(F) but it does not depend on the size of the representation. As a consequence of this fact, the same dichotomy holds for representation classes which, in terms of absolute expressivity power, are equivalent to formulas, such as boolean circuits.

9.5.2 An Extended Notion of pwm-reduction

To derive some results in this section we need to generalize slightly the notion of pwm-reduction presented in Definition 2. (See Section 2.3.2 for extended details about learning concepts).

Definition 27 Let $\mathcal{C}$ and $\mathcal{C}'$ be representations of concepts. Let $\perp$ and $\top$ be elements not in X. Then $\mathcal{C}$ is *epwm-reducible* to $\mathcal{C}'$, denoted $\mathcal{C} \leq_{\text{epwm}} \mathcal{C}'$, if and only if there exist four mappings g,f,h, and j 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 $x \in X$ 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 $|u| \leq s$, for every $x' \in X$, and for every $b \in \{\top,\perp\}$, $h(s,n,x')$ is a string $x \in X$, and $j(s,n,x',b)$ is either $\perp$ or $\top$. Furthermore $x' \in K_{\mathcal{C}'}(g(s,n,u))$ if and only if $j(s,n,x',b) = \top$, where $b = \top$ if $x \in K_{\mathcal{C}}(u)$ and $b = \perp$ otherwise. Moreover, h and j are 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 "$x \notin K_{\mathcal{C}}(u)$".

This definition is obtained from the definition of pwm-reduction (Definition 2) by replacing in (3), the expression "$x' \in K_{\mathcal{C}'}(g(s,n,u))$ if and only if $x \in K_{\mathcal{C}}(u)$" by "$x' \in K_{\mathcal{C}'}(g(s,n,u))$ if and only if $j(s,n,x',b) = \top$, where $b = \top$ if $x \in K_{\mathcal{C}}(u)$ and $b = \bot$ otherwise". The intuition underlying this new definition is the following: $\mathcal{C} \leq_{\text{epwm}} \mathcal{C}'$ means that $\mathcal{C}$ can be predicted with membership queries using an algorithm A that predicts $\mathcal{C}'$ with membership queries as an oracle. When algorithm A asks a membership query, i.e., an string x' , condition (3) in the definition of pwm-reduction guarantees that if is possible to find an string $x = h(s,n,x')$ in polynomial time and provide A with the answer of the membership query with sting x. The definition of the epwm-reduction is more flexible in the sense that it allows, for example, to answer the membership query with x' directly. It allows also to anwer the membership query with string x' with some function that depends on the value of the result of the membership query over x, for example, its boolean negation.

Finally, notice that the definition of pwm-reduction is obtained by forcing $j(s,n,x',b) = b$ in the definition of epwm-reduction.

The following results are obtained adapting slightly some proofs in [3].

Lemma 44 *The epwm-reduction is transitive, i.e., let $\mathcal{C}, \mathcal{C}'$ and $\mathcal{C}''$ be representations of concepts, if $\mathcal{C} \leq_{epwm} \mathcal{C}' \leq_{epwm} \mathcal{C}''$ then $\mathcal{C} \leq_{epwm} \mathcal{C}''$.*

Lemma 45 *Let $\mathcal{C}$ and $\mathcal{C}'$ be representations of concepts. If $\mathcal{C} \leq_{epwm} \mathcal{C}'$ and $\mathcal{C}'$ is polynomially predictable with membership queries, then $\mathcal{C}$ is also polynomially predictable with membership queries.*

9.5.3 The Dichotomy Theorem

A *basis* F is a finite set of boolean functions $\{f_1, f_2, \ldots, f_n\}$ (f_i $1 \leq i \leq n$, denotes both the function and its symbol). We follow the standard definitions of boolean circuits and boolean formulas. The class of boolean circuits over the basis F, denoted by CIR(F), is defined to be the set of all the boolean circuits where every gate is a function in F. The class of boolean formulas over the basis F, denoted by FOR(F) is the class of circuits in CIR(F) with fan-out ≤ 1.

Given a boolean circuit C over the input variables $x_1, x_2, \ldots, x_n$, we denote by $[C]$ the function computed by C when the variables are taken as arguments in lexicographical order. In the same way, given a finite set of boolean functions F we define $[\mathrm{CIR}(F)]$ as the class of functions computed by circuits in $\mathrm{CIR}(F)$. Similarly, given a boolean formula Φ over the variables $x_1, x_2, \ldots, x_n$, we denote by $[\Phi]$ the function computed by Φ when the variables are taken as arguments in lexicographical order. In the same way, given a finite set of boolean functions F we define $[\mathrm{FOR}(F)]$ as the class of functions computed by formulas in $\mathrm{FOR}(F)$. Given a boolean circuit $C \in \mathrm{CIR}(F)$, it is possible to construct a formula $\Phi \in \mathrm{FOR}(F)$ computing function computed by C (the size of Φ can be exponentially bigger than the size of C but we are not concerned with the size of the representation). Thus $[\mathrm{FOR}(F)] = [\mathrm{CIR}(F)]$.

In fact, it is direct to verify that the class of functions $[\mathrm{CIR}(F)]$ contains the projections and it is closed under composition. Therefore, it constitutes a clone. More precisely, $[\mathrm{CIR}(F)]$ is exactly the clone generated by F, in our terminology

$$[\mathrm{CIR}(F)] = [\mathrm{FOR}(F)] = \langle F \rangle_o, \text{ for all bases } F.$$

The use of clone theory to study computational problems over boolean formulas was introduced in [79] studying the complexity of some computational problems over boolean formulas, such as satisfiability, tautology and some counting problems.

We say that an n-ary function f is disjunctive if there exists some boolean coefficients c_i $(0 \leq i \leq n)$ such that

$$f(x_1, \ldots, x_n) = c_0 \vee (c_1 \wedge x_1) \vee (c_2 \wedge x_2) \vee \cdots \vee (c_n \wedge x_n).$$

Similarly, we say that an n-ary function f is conjunctive if there exists some boolean coefficients c_i $(0 \leq i \leq n)$ such that

$$f(x_1, \ldots, x_n) = c_0 \wedge (c_1 \vee x_1) \wedge (c_2 \vee x_2) \wedge \cdots \wedge (c_n \vee x_n).$$

Accordingly, we say that an n-ary function f is linear if there exists some boolean coefficients c_i $(0 \leq i \leq n)$ such that

$$f(x_1, \ldots, x_n) = c_0 \oplus (c_1 \wedge x_1) \oplus (c_2 \wedge x_2) \oplus \cdots \oplus (c_n \wedge x_n).$$

Let F be a set of boolean functions. We will say that F is disjunctive (resp. conjunctive, linear) iff every function in F is disjunctive (resp. conjunctive, linear). We will say that F is basic iff F is disjunctive, conjunctive or linear.

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

In this section we state and prove the main result of the paper.

Theorem 54 *(Dichotomy Theorem for the Learnability of Boolean Circuits and Boolean Formulas) Let F be a finite set of boolean functions. If F is basic, then $\mathcal{C}_{\mathrm{CIR}(F)}$ is both polynomially exactly learnable with $n+1$ equivalence queries, and polynomially exactly learnable with $n+1$ membership queries, where n is the number of variables. Otherwise, $\mathcal{C}_{\mathrm{FOR}(F)}$ is not polynomially predictable with membership queries under the assumption that any of the following three problems are intractable: testing quadratic residues modulo a composite, inverting RSA encryption, or factoring Blum integers.*

We refer the reader to Angluin and Kharitonov [3] for definitions of the cryptographic concepts. We mention that the non-learnability results for boolean circuits hold also with the weaker assumption of the existence of public-key encryption systems secure against CC-attack, as discussed in [3].

Proof of Theorem 54

Learnability of formulas over disjunctive, conjunctive and linear bases is rather straightforward. Notice that if a basis F is disjunctive (resp. conjunctive, linear) then every function computed by a circuit in $\mathrm{CIR}(F)$ is also disjunctive (resp. conjunctive, linear). Thus, learning a boolean circuit over a basic basis is reduced to finding the boolean coefficients of the canonical expression. It is easy to verify that this task can be done in polynomial time with $n+1$ equivalence queries or $n+1$ membership queries, as stated in the theorem.

Now, we study which portion of Post's lattice is covered by these cases. Consider clone L_1 with generating set $L_1 = \langle \varphi_{O_4}, \varphi_{O_5}, \varphi_{O_6}, \varphi_{L_4}\rangle_o$. Since all the operations in the generating set of L_1 are linear, then the clone L_1 is linear. Similarly, clone P_6, generated by operations φ_{O_5}, φ_{O_6} and φ_{P_1}

is conjunctive, since so is every operation in the generating set. Finally clone S_6, generated by operations φ_{O_5}, φ_{O_6} and φ_{S_1} is disjunctive, since so is every operation in the generating set. Thus, every clone contained in L_1, P_6, or S_6 is linear, conjunctive or disjunctive respectively. Actually, clone L_1 (resp. P_6, S_6) has been chosen carefully among the linear (resp. conjunctive, disjunctive) clones. It corresponds to the maximal linear (resp. conjunctive, disjunctive) clone in Post's lattice. It has been obtained as the join of all the linear (resp. conjunctive, disjunctive) clones and, in consequence, has the property that every linear (resp. conjunctive, disjunctive) clone is contained in L_1 (resp. P_6, S_6).

Let us study non-basic clones. With a simple inspection of Post's lattice we can infer that any clone not contained in L_1, S_6 or P_6 contains any of the following operations: $\varphi_{F_2^\infty}$, $\varphi_{F_6^\infty}$, φ_{D_2}. In Section 9.5.4 it is proved that if $\langle F \rangle_o$ contains any of the previous functions, then the class FOR(F) is not PAC-predictable even with membership queries under the assumptions of the theorem. ∎

9.5.4 Three Fundamental Non-Learnable Functions

Let $\mathcal{B} = \{\text{AND}, \text{OR}, \text{NOT}\}$ be the usual complete basis for boolean formulas. In [3], it is proved that $\mathcal{C}_{\text{FOR}(\mathcal{B})}$ is not polynomially predictable under the assumptions of Theorem 54. In this section we generalize the previous result to all bases F able to "simulate" any of the following three basic non-learnable functions: $\varphi_{F_2^\infty}$, $\varphi_{F_6^\infty}$, and φ_{D_2}. This three functions can be regarded as the basic causes for non-learnability in formulas.

The technique used to prove non-learnability results is a two-stage pwm-reduction from FOR($\mathcal{B}$). First, we prove as an intermediate result that monotone boolean formulas are as hard to learn, as general boolean formulas.

Lemma 46 *The class $\mathcal{C}_{\text{FOR}(\mathcal{B})}$ is pwm-reducible to $\mathcal{C}_{\text{FOR}(\{\text{AND},\text{OR}\})}$.*

Proof. This proof mimics the proof of Lemma 15 in Chapter 3. Let Φ be a boolean formula with $x_1, \ldots, x_n$ as input variables. We can assume that NOT operations are applied only to input variables. Otherwise by using repeatedly De Morgan's law we can move every NOT function towards the input variables.

Consider the formula Ψ with $x_1, \ldots, x_n, y_1, \ldots, y_n$ as input variables, obtained modifying slightly formula Φ as follows: replace every NOT function applied to variable x_i, by the new input variable y_i. Finally, we define the formula Υ with $x_1, \ldots, x_n, y_1, \ldots, y_n$ as input variables to be:

$$\Upsilon(x_1, \ldots, x_n, y_1, \ldots, y_n) = \left(\Psi(x_1, \ldots, x_n, y_1, \ldots, y_n) \vee \bigvee_{1 \le i \le n} (x_i \wedge y_i)\right) \wedge \wedge \bigwedge_{1 \le i \le n} (x_i \vee y_i)$$

Formula Υ evaluates the following function:

$$\Upsilon(x_1, \ldots, x_n, y_1, \ldots, y_n) = \begin{cases} \Phi(x_1, \ldots, x_n) & \text{if } \forall i : 1 \le i \le n,\ x_i \ne y_i \\ 0 \quad \text{if } \exists i : 1 \le i \le n,\ x_i = y_i = 0 \\ 1 \quad \text{otherwise} \end{cases}$$

For every natural number s and for every concept representation $u \in \mathcal{C}_{\mathrm{FOR}(\mathcal{B})}$ such that $|u| \le s$, let Φ be the boolean formula with n input variables represented by u, let u' be the representation of the monotone boolean formula Υ obtained from Φ as described above. We define $g(s, n, u) = u'$. For every assignments x and y of length n we define $f(s, n, x) = x\overline{x}$, $h(s, n, xy) = x$ and

$$j(s, n, xy, b) = \begin{cases} b & \text{if } x = \overline{y} \\ \bot & \text{if } \exists i : 1 \le i \le n, x_i = y_i = 0 \\ \top & \text{otherwise} \end{cases}$$

Clearly, f, g, h and j satisfy the conditions (1), (2) and (3) in Definition 27 and therefore define a pwm-reduction.

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

Now, we have to see that $\mathcal{C}_{\mathrm{FOR}(\{\mathrm{NOT,AND}\})}$ is pwm-reducible to $\mathcal{C}_{\mathrm{FOR}(F)}$ if F is able to "generate" any of the following functions: $\varphi_{F_2^\infty}$, $\varphi_{F_6^\infty}$ or φ_{D_2}. Let us do a case analysis.

Theorem 55 *Let F be a set of boolean functions. If $\varphi_{F_2^\infty} \in \langle F \rangle_o$ then $\mathcal{C}_{\mathrm{FOR}(\{\mathrm{AND,OR}\})}$ is pwm-reducible to $\mathcal{C}_{\mathrm{FOR}(F)}$.*

Proof. Clone $\langle F\rangle_o$ includes the operation $\varphi_{F_2^\infty}(x,y,z) = x \vee (y \wedge z)$. Thus, there exists some formula $\Phi_{F_2^\infty}$ in FOR(F) over three variables x, y, z such that $[\Phi_{F_2^\infty}] = \varphi_{F_2^\infty}$.

Clearly, with the operation $\varphi_{F_2^\infty}$ and the additional help of constant 0 it is possible to simulate functions AND and OR,

$$\mathrm{AND}(x,y) = \varphi_{F_2^\infty}(0,x,y)$$
$$\mathrm{OR}(x,y) = \varphi_{F_2^\infty}(x,y,y)$$

Let Ψ be an arbitrary monotone boolean formula in FOR($\{\mathrm{AND},\mathrm{OR}\}$) over the input variables $x_1, x_2 \ldots, x_n$.

Let Υ_2 be the boolean formula over the input variables $x_1, \ldots, x_n, c_0$ obtained from Ψ by replacing every occurrence AND(x,y) by $\Phi_{F_2^\infty}(c_0,x,y)$ and, similarly, every occurrence of OR(x,y) by $\Phi_{F_2^\infty}(x,y,y)$.

Finally, let Υ_1 be the boolean formula defined by:

$$\Upsilon_1(x_1,x_2,\ldots,x_n,c_0) = \Phi_{\Upsilon_2^\infty}(\Upsilon_2(x_1,x_2\ldots,x_n,c_0),c_0,c_0),$$

By construction we have

$$\Upsilon_2(x_1,x_2,\ldots,x_n,0) = \Psi(x_1,x_2,\ldots,x_n), \text{and}$$
$$\Upsilon_1(x_1,x_2,\ldots,x_n,c_0) = \begin{cases} \Psi(x_1,x_2,\ldots,x_n) & \text{if } c_0 = 0 \\ 1 & \text{otherwise} \end{cases}$$

Now we are in position to define the pwm-reduction. Let g be the function assigning to every monotone boolean formula Ψ, an associated formula Υ_1 in FOR(F) constructed as described above. Let f be the function adding the value of the constant zero to the end of string, i.e.,

$$f(s,n,\langle x_1,x_2,\ldots,x_n\rangle) = \langle x_1,x_2,\ldots,x_n,0\rangle.$$

Mapping h produces the inverse result. That is, given an string removes the last value (corresponding to the constant 0).

$$h(s,n,\langle x_1,\ldots,x_n,c_0\rangle) = \langle x_1,\ldots,x_n\rangle$$

Finally, function j is defined by

$$j(s,n,\langle x_1,\ldots,x_n,c_0\rangle,b) = \begin{cases} b & \text{if } c_0 = 0 \\ \top & \text{otherwise} \end{cases}$$

Thus, it is immediate to verify that f, g, h, and j define a pwm-reduction from $\mathcal{C}_{\{\mathrm{AND},\mathrm{OR}\}}$ to $\mathcal{C}_{\mathrm{FOR}(F)}$. ■

By duality we have,

Theorem 56 *Let F be a set of boolean functions. If $\varphi_{F_6^\infty} \in \langle F\rangle_o$ then $\mathcal{C}_{\mathrm{FOR}(\{\mathrm{AND},\mathrm{OR}\})}$ is pwm-reducible to $\mathcal{C}_{\mathrm{FOR}(F)}$.*

Finally we study clones containing φ_{D_2}.

Theorem 57 *Let F be a set of boolean functions. If $\varphi_{D_2} \in \langle F\rangle_o$ then $\mathcal{C}_{\mathrm{FOR}(\{\mathrm{AND},\mathrm{OR}\})}$ is pwm-reducible to $\mathcal{C}_{\mathrm{FOR}(F)}$.*

Proof. For this proof we will use the fact that operation φ_{D_2} satisfies the self-duality property. Thus, every function in $\langle\varphi_{D_2}\rangle_o$ is self-dual. Clone $\langle F\rangle_o$ includes the majority operation $\varphi_{D_2}(x,y,z) = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$. Thus, there exists some formula Φ_{D_2} in FOR(F) over three variables x, y, z such that the function computed by Φ_{D_2} is φ_{D_2}.

Clearly, with the operation φ_{D_2} and the additional help of constants it is possible to simulate functions AND and OR,

$$\mathrm{AND}(x,y) = \varphi_{D_2}(0,x,y)$$

$$\mathrm{OR}(x,y) = \varphi_{D_2}(1,x,y)$$

For every monotone boolean formula Ψ in FOR($\{\mathrm{AND},\mathrm{OR}\}$) over the input variables $x_1, x_2 \ldots, x_n$ we construct an associated formula Υ_1 over the variables $x_1, x_2 \ldots, x_n, c_0, c_1$ defined by

$$\Upsilon_1(x_1,x_2,\ldots,x_n,c_0,c_1) = \Phi_{D_2}(\Upsilon_2(x_1,x_2\ldots,x_n,c_0,c_1),c_0,c_1),$$

where Υ_2 is the formula over the input variables $x_1, x_2, \ldots, x_n, c_0, c_1$ obtained from Ψ in a similar way to the previous proof: we replace every occurrence of AND(x,y) by $\Phi_{D_2}(c_0,x,y)$ and every occurrence of OR(x,y) by $\Phi_{D_2}(c_1,x,y)$.

By construction we have (the case $c_0 = 1 \wedge c_1 = 0$ is a consequence of the self-duality of D_2),

$$\Upsilon_2(x_1,x_2,\ldots,x_n,c_0,c_1) = \begin{cases} \Psi(x_1,x_2,\ldots,x_n) & \text{if } c_0 = 0 \wedge c_1 = 1 \\ \neg\Psi(\neg x_1,\neg x_2,\ldots,\neg x_n) & \text{if } c_0 = 1 \wedge c_1 = 0 \\ \text{undetermined} & \text{otherwise} \end{cases}$$

$$\Upsilon_1(x_1,x_2,\ldots,x_n,c_0,c_1) = \begin{cases} 0 & \text{if } c_0 = c_1 = 0 \\ 1 & \text{if } c_0 = c_1 = 1 \\ \Psi(x_1,x_2,\ldots,x_n) & \text{if } c_0 = 0 \wedge c_1 = 1 \\ \neg\Psi(\neg x_1,\neg x_2,\ldots,\neg x_n) & \text{if } c_0 = 1 \wedge c_1 = 0 \end{cases}$$

Now we are in position to define the pwm-reduction. Let g be the function assigning to every monotone boolean formula Ψ in $\mathrm{FOR}(\{\mathrm{AND}, \mathrm{OR}\})$, an associated formula Υ_1 in $\mathrm{FOR}(F)$ constructed as described above. Let f be the function adding the value of the constants to the end of string, i.e.,

$$f(s, n, \langle x_1, x_2, \ldots, x_n \rangle) = \langle x_1, x_2, \ldots, x_n, 0, 1 \rangle.$$

Mapping h removes the last two values (corresponding to the constants) and, moreover, h negates the values of the assignment if the values of the constants are flipped,

$$h(s, n, \langle x_1, \ldots, x_n, c_0, c_1 \rangle) = \begin{cases} \langle x_1, \ldots, x_n \rangle & \text{if } c_0 = 0 \vee c_1 = 1 \\ \langle \neg x_1, \ldots, \neg x_n \rangle & \text{otherwise} \end{cases}$$

Finally, function j is defined by

$$j(s, n, \langle x_1, \ldots, x_n, c_0, c_1 \rangle, b) = \begin{cases} \bot & \text{if } c_0 = 0 \wedge c_1 = 0 \\ b & \text{if } c_0 = 0 \wedge c_1 = 1 \\ \neg b & \text{if } c_0 = 1 \wedge c_1 = 0 \\ \top & \text{if } c_0 = 1 \wedge c_1 = 1 \end{cases}$$

Thus, it is immediate to verify that f, g, h, and j define a pwm-reduction from $\mathcal{C}_{\{\mathrm{AND},\mathrm{OR}\}}$ to $\mathcal{C}_{\mathrm{CIR}(F)}$. ■

Chapter 10

Conclusion and Open Problems

10.1 Conclusion

We studied the satisfiability and learnability of generalized quantified Boolean formulas, completing the knowledge about the subject. We also generalized these results to higher domains. For the satisfiability problem we characterized, in terms of closure operations, the class of subgroup and width 1 problems, completing the research project started in [48]. We also identified a new class of tractable problems called para-primal problems, incomparable with the all the previously identified classes. Finally, we started the study of the learnability of quantified formulas over arbitrary (non-necessarily boolean) domains by identifying three families of polynomially learnable formulas, containing, as a particular case, all the efficiently learnable formulas over the Boolean domain.

10.2 Open Problems

The most interesting remaining problem is to complete the dichotomic classification for the tractability of constraint satisfaction problems over a fixed basis. Up to the present moment, four families of tractable subclasses have been identified, namely, near-unanimity, coset generating, set function and para-primal problems. Recall that, the approach based in the relationship between the constraint satisfaction problems and clones in universal algebra is limited by the fact that the structure of the clone lattice for domains with more than two elements is largely unknown. The first step in this long-term project would be to study the tractability of the algebras satisfying the following interesting property that come out in the study of the learnability of quantified formulas and that was also pointed out in [31].

Definition 28 An algebra $\mathbf{D}$ is *polynomially generated* iff every subuniverse of $\mathbf{D}^n$ is generated by a number of elements polynomial in n.

Our study is based in the following conjecture:

Conjecture 1 *Every polynomially generated algebra is tractable*

Near-unanimity operations, coset generating operations and para-primal algebras are examples of such feature. This property also guarantees that an algebra is polynomial-time learnable in the model of exact learning with equivalence queries. Based on this fact we could attempt to find a complete classification for the learnability of quantified boolean formulas proving the following conjecture:

Conjecture 2 *An algebra is polynomial-time learnable iff it is polynomially generated.*

It is also very important to characterize, in terms of polymorphisms, the polynomially generated algebras.

Another interesting open problem raises from the following consideration: every subclass of constraint satisfaction problems presented in this thesis is obtained by fixing the set of basic constraint relations. It is also possible to define interesting families of constraint satisfaction problems by restricting the possible ways in which variables can be constrained, that is, the constraint interconnection hypergraph. Despite the large amount of individual results about the tractability of such families, a uniform study of such problems is not known.

We plan to study this problem with the following approach: to convert a constraint satisfaction problem with some restriction in the hypergraph interconnection into a constraint satisfaction problem over a fixed basis in order to use the core of knowledge about clones to deduce tractability conditions.

Bibliography

[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] B. Aspvall, M. Plass, and R. Tarjan. A Linear-time Algorithm for Testing the Truth of Certain Quantified Boolean Formulas. *Information Processing Letters*, 8:121–123, 1979.

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

[6] U. Berggren. Linear Time Deterministic Learning of k-term DNF. In *6th Annual ACM Conference on Computational Learning Theory, COLT'93*, pages 37–40, 1993.

[7] V.G. Bodnarchuk, L.A. Kaluzhnin, V.N. Kotov, and B.A. Romov. Galois Theory for Post Algebras. I,II. *Cybernetics*, 5:243–252, 531–539, 1969.

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

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

[10] D.M. Clark and P.H. Krauss. Para Primal Algebras. *Algebra Universalis*, 6:165–192, 1976.

[11] D.M Clark and P.H. Krauss. Plain Para Primal Algebras. *Algebra Universalis*, 11:365–388, 1980.

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

[13] S.A. Cook. The Complexity of Theorem-Proving Procedures. In *3rd Annual ACM Symposium on Theory of Computing STOC'71*, pages 151–158, 1971.

[14] M.C. Cooper. An Optimal k-consistency Algorithm. *Artificial Intelligence*, 41:89–95, 1989.

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

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

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

[18] V. Dalmau. A Dichotomy Theorem for Learning Quantified Boolean Formulas. In *10th Annual ACM Conference on Computational Learning Theory COLT'97*, pages 193–200, 1997.

[19] V. Dalmau. Some Dichotomy Theorems on Constant-free Quantified Boolean Formulas. Technical Report LSI–97–43–R, Departament LSI, Universitat Politècnica de Catalunya, 1997.

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

[21] V. Dalmau. Boolean Formulas are Hard to Learn for Most Gate Basis. In *10th International Conference on Algorithmic Learning Theory, ALT'99*, volume 1720 of *Lecture Notes in Artificial Intelligence*, pages 301–312, Berlin/New York, 1999.

[22] V. Dalmau. A New Tractable Class of Constraint Satisfaction Problems. In *6th International Symposium on Artificial Intelligence and Mathematics*, 2000.

[23] V. Dalmau and P. Jeavons. Learnability of Quantified Formulas. In *4th European Conference on Computational Learning Theory Eurocolt'99*, volume 1572 of *Lecture Notes in Artificial Intelligence*, pages 63–78, Berlin/New York, 1999. Springer-Verlag.

[24] V. Dalmau and J. Pearson. Set Functions and Width 1. In *5th International Conference on Principles and Practice of Constraint Programming, CP'99*, volume 1713 of *Lecture Notes in Computer Science*, pages 159–173, Berlin/New York, 1999. Springer-Verlag.

[25] R. Dechter. From Local to Global Consistency. *Artificial Intelligence*, 55:87–107, 1992.

[26] R. Dechter and J. Pearl. Network-based Heuristics for Constraint Satisfaction Problems. *Artificial Intelligence*, 34(1):1–38, 1988.

[27] R. Dechter and J. Pearl. Structure Identification in Relational Data. *Artificial Intelligence*, 58:237–270, 1992.

[28] R. Dechter and P. van Beek. Local and Global Relational Consistency. *Theoretical Computer Science*, 173:283–308, 1997.

[29] W. Dowlin and J. Gallier. Linear-time Algorithms for Testing the Satisfiability of Propositional Horn Formulae. *Journal of Logic Programming*, 1(3), 1984.

[30] O. Ekin, S. Foldes, P.L. Hammer, and L. Hellerstein. Equational Theories of Boolean Functions. Technical Report 97–79, DIMACS, December 1997. To appear in Discrete Mathematics.

[31] 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.

[32] A. Feigelson and L. Hellerstein. The Forbidden Projections of Unate Functions. *Discrete Applied Mathematics*, 77(3):221–236, 1997.

[33] E.C. Freuder. Synthesizing Constraint Expressions. *Comm. ACM*, 21:958–966, 1978.

[34] E.C. Freuder. A Sufficient Condition for Backtrack-free Search. *Journal of the ACM*, 29:24–32, 1982.

[35] E.C. Freuder. A Sufficient Condition for Backtrack-bounded Search. *Journal of the ACM*, 32:755–761, 1985.

[36] D. Geiger. Closed Systems of Functions and Predicates. *Pac. J. Math.*, 27(2):228–250, 1968.

[37] Jens-Uwe Grabowski and Reinhard Pöschel. Rigid Relational Systems: A General Approach. *Period. Math. Hung.*, 32(1–2):35–44, 1996.

[38] R. Greenlaw, H.J. Hoover, and W.L. Ruzzo. *Limits to Parallel Computation: P-completeness theory*. Oxford University Press, 1995.

[39] M. Gyssens, P. Jeavons, and D. Cohen. Decomposing Constraint Satisfaction Problems using Database Techniques. *Artificial Intelligence*, 66(1):57–89, 1994.

[40] D. Haussler, N. Littlestone, and M.K. Warmuth. Predicting $\{0, 1\}$ Functions on Randomly Drawn Points. *Information and Computation*, 115(2):248–292, 1994.

[41] P. Hell and J. Nešetřil. On the Complexity of H-coloring. *J. Comb. Theory, Series B*, 48:92–110, 1990.

[42] L. Hellerstein. On Generalized Constraints and Certificates. Technical Report 26–98, RUTCOR, September 1998.

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

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

[45] A. Horn. On Sentences wich are True of Direct Unions of Algebras. *Journal of Simbolic Logic*, 16:14–21, 1951.

[46] 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.

[47] P. Jeavons. On the Algebraic Structure of Combinatorial Problems. *Theoretical Computer Science*, 200:185–204, 1998.

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

[49] 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, Berlin/New York, 1995. Springer-Verlag.

[50] P. Jeavons, D. Cohen, and M. Gyssens. A Test for Tractability. In *2nd International Conference on Principles and Practice of Constraint Programming CP'96*, volume 1118 of *Lecture Notes in Computer Science*, pages 267–281, Berlin/New York, August 1996. Springer-Verlag.

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

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

[53] M. Karpinski, M. Kleine Büning, and P. Schmitt. On the Computational Complexity of Quantified Horn Clauses. In *Computer Science Logic '87*, pages 129–137, 1987.

[54] D. Kavvadias and M. Sideri. The Inverse Satisfiability Problem. In *2nd Computing and Combinatorics COCOON'96*, volume 1090 of *Lecture Notes in Computer Science*, pages 250–259, Berlin/New York, 1996. Springer-Verlag.

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

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

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

[58] R. Khardon. Learning Function Free Horn Expressions. In *11th Annual ACM Conference on Computational Learning Theory*, 1998.

[59] L. Kirousis. Fast Parallel Constraint Satisfaction. *Artificial Intelligence*, 64:147–160, 1993.

[60] 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.

[61] A. K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8:99–118, 1977.

[62] S.S. Marchenkov. Homogeneous Algebras. *Problemy Kibernetiki*, 39:85–106, 1982.

[63] E. Marczewski. Homogeneous Algebras and Homogeneous Operations. *Fund. Math.*, 56:81–103, 1964.

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

[65] U. Montanari. Networks of Constraints: Fundamental Properties and Applications to Picture Processing. *Information Sciences*, 7:95–132, 1974.

[66] U. Montanari and F. Rossi. Constraint Relaxation may be Perfect. *Artificial Intelligence*, 48:143–170, 1991.

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

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

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

[70] C. H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[71] K. Pillaipakkamnat and V. Raghavan. A Linear Time Equivalence Test for Read-twice DNF Formulas. *Information Processing Letters*, 48:289–295, 1993.

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

[73] N. Pippenger. Galois Theory for Minors of Finite Functions. Technical Report 98-08, Department of Computer Science, University of British Columbia, 1998.

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

[75] R. Pöschel and L.A. Kalužnin. *Funktionen- und Relationenalgebren.* VEB Deutscher Verlag der Wissenschaften, Birkhäuser Verlag, Basel and Stuttgart, 1979.

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

[77] R.W. Quackenbush. Minimal Para Primal Algebras. In *Contributions to General Algebra 2, Proc. Klagenfurt Conf. 1982*, pages 291–304, 1983.

[78] S. Reith and H. Vollmer. The Complexity of Computing Optimal Assignments of Generalized Propositional Formulae. Technical Report TR196, Department of Computer Science, Universität Würzburg, 1999.

[79] S. Reith and K.W. Wagner. The Complexity of Problems Defined by Subclasses of Boolean Functions. Technical Report TR218, Department of Computer Science, Universität Würzburg, 1999.

[80] 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.

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

[82] L. Stockmeyer. The Polynomial-time Hierarchy. *Theoretical Computer Science*, 3:1–22, 1976.

[83] L. Stockmeyer and A. Meyer. Word Problems Requiring Exponential Time: Preliminary Report. In *5th Annual ACM Symposium on Theory of Computing*, pages 1–9, 1973.

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

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

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

[87] P. van Beek and R. Dechter. On the Minimality and Decomposability of Row-convex Constraint Networks. *Journal of the ACM*, 42:543–561, 1995.

[88] P. van Hentenryck, Y. Deville, and C-M. Teng. A Generic Arc-consistency Algorithm and its Specializations. *Artificial Intelligence*, 1992.

[89] M. Ward. The Closure Operators of a Lattice. *Ann. of Math.*, 43:191–196, 1942.