Quantified Constraint Satisfaction Problems: Closure Properties, Complexity, and Algorithms

Hubie Chen*

Abstract

There has been much prior work on understanding the complexity of the constraint satisfaction problem (CSP), a broad framework capturing many combinatorial problems. This paper studies the complexity of the quantified constraint satisfaction problem (QCSP), a natural and strict generalization of the CSP. The CSP can be viewed as the problem of deciding whether or not existentially quantified formulas of a certain form are true; in the QCSP, universal quantification is permitted in addition to existential quantification. Our main contribution is the identification of three broad tractable subclasses of the QCSP. Central to our study is the algebraic notion of closure properties of constraints, which has been previously used to study the CSP.

1 Introduction

The *constraint satisfaction problem* (CSP) is a general framework in which many combinatorial problems can be expressed. An instance of a CSP consists of a network of constraints over a set of variables; the problem is to decide whether or not there is an assignment to the variable set satisfying all of the constraints. A natural and strict generalization of the CSP is the *quantified constraint satisfaction problem* (QCSP). An instance of a QCSP can be viewed as a game between two players which alternate setting variables of a constraint network; the problem is to decide whether or not a specified player can always succeed in satisfying all of the constraints, despite the moves of the other player. While the CSP captures the complexity of deciding whether or not a combinatorial object of desirable type exists in a large search space, the QCSP captures the complexity of many problems involving *interaction* among multiple agents. Such problems arise naturally in a wide variety of domains, for example, artificial intelligence, logic, and game theory.

In general, the CSP and QCSP are intractable, being NP-complete and PSPACE-complete, respectively. A popular and oft-taken path for coping with computationally hard problems is the *identification of polynomial-time tractable subclasses*. There are multiple ways to place restrictions on the CSP and QCSP in hopes of obtaining such tractable subclasses [23]. One way is to restrict the *constraint language*, or the types of constraints that are permitted in instances. This is the form of restriction which was studied by Schaefer in his now classic dichotomy theorem [27], and has seen intense investigation over the past decade in several different contexts [10]. In this paper, we study the complexity of the QCSP from the standpoint of restricting the constraint language. Our main contribution is the identification of three broad tractable subclasses of the QCSP.

1.1 Background

In [27], Schaefer proved a dichotomy theorem on the complexity of CSP, namely, that every constraint language over a two-element domain gives rise to a CSP subclass which is either in P or is NP-complete.

*Department of Computer Science, Cornell University, Ithaca, NY 14853. Email: `hubes@cs.cornell.edu`.

The non-trivial tractable subclasses given by this result are 2-SAT, HORN SAT, and XOR-SAT (where each constraint is a linear equation in the field with two elements). Over the past decade, many more complexity dichotomy theorems in the spirit of Schaefer's have been established, including dichotomies in the complexity of counting the number of satisfying assignments [9], finding the lexicographically minimum satisfying assignment [26], "inverse" satisfiability [20], and finding a satisfying assignment maximizing the number of satisfied constraints [21]. All of these dichotomy theorems are for constraint languages over a two-element domain. Particularly relevant here is the dichotomy theorem for QCSP [10, 11] (also for two-element domains), which shows that the only tractable subclasses in this context are QUANTIFIED 2-SAT [1], QUANTIFIED HORN SAT [8], and QUANTIFIED XOR-SAT [10], reflecting exactly the non-trivial tractable subclasses of CSP given by Schaefer's theorem. All other constraint languages give rise to a PSPACE-complete QCSP subclass.

These classification theorems are valuable because they identify *all* of the ways tractability can arise from restrictions on the constraint language. In addition, knowledge of intractable subclasses can be quite useful in developing further proofs of intractability. Thus, a research program of great importance is to extend the existing classification theorems, such as Schaefer's theorem, to constraint languages over any finite domain. Much attention has recently been directed towards accomplishing this in the case of the CSP, and has resulted in the identification of many new tractable subclasses of the CSP [24, 14, 18, 22, 3, 12, 7, 5, 13]. A phenomenal result produced by this strain of work is Bulatov's dichotomy theorem on CSP complexity classifying all constraint languages over a *three*-element domain [4], which resolved a question from Schaefer's original paper [27] that had been open for over two decades.

An *algebraic approach* to studying the relative expressiveness of constraint languages, which we adopt here to study the QCSP, was introduced in [24] and further studied in [17, 18, 19, 6, 3, 7, 4, 5, 12]. In this approach, a dual perspective on constraint languages is given by studying the set of functions under which a constraint language is closed, or the *closure properties* of a constraint language. In particular, closure properties are linked to constraint languages by a *Galois connection*, and two different constraint languages having the same closure properties give rise to subclasses of the CSP (and QCSP) with *exactly* the same complexity. The program of classifying constraint languages as either "tractable" or "intractable" can thus be rephrased as a classification question on closure properties; as it turns out, this rephrasing makes the classification program amenable to attack by insights and tools from universal algebra. This dual viewpoint was used heavily by Bulatov to obtain his dichotomy theorem [4], and can also be used in conjunction with Post's classification theorem [25] to succinctly derive Schaefer's theorem [2]. (The detailed combinatorics on contraint languages performed by Schaefer are effectively transferred over to the domain of functions, making Post's theorem immediately applicable.)

1.2 Results in this paper

In this paper, we take the initial steps towards the long-term goal of understanding the complexity of quantified constraint satisfaction in domains of size greater than two. We first review the basic terminology of constraint satisfaction, and then discuss how the "closure properties" approach to studying constraint satisfaction can be adapted to study quantified constraint satisfaction (Section 2). We then give a description of the tractable QCSP subclasses (for two-element domains) in terms of Schaefer's description of the tractable CSP subclasses (Section 3). This description demonstrates, in a precise sense, that the tractable QCSP subclasses reflect exactly the tractable CSP subclasses, in domain size two.

We then give three proof systems for QCSPs; these provide mechanisms for deriving "implied" constraints from a given QCSP instance (Section 4). The first proof system is sound and complete in the sense that an "empty constraint" is derivable from a QCSP instance if and only if the instance is unsatisfiable. The third proof system is sound and complete for the class of QCSPs with constraint language closed under a semilattice operation, and the second proof system is intermediate in generality between the first and third.

Our proof systems can be viewed as generalizations of the *Q-resolution* and *Q-unit-resolution* proof systems given in [8], which are defined only for domain size two QCSPs having a constraint network given in conjunctive normal form.

Next, we show three sufficient conditions for a constraint language to give a tractable QCSP subclass – namely, closure under a semilattice operation with bottom, closure under a coset operation, and closure under a near-unanimity operation (Section 5). To establish the first of these tractability results, we make direct use of our third proof system. These three tractability results generalize and unify two different groups of previously established results. Our results broaden the three tractable QCSP subclasses in domain size two named above, by showing that they are merely instances of larger tractable subclasses which exist for domains of all sizes. In addition, the three closure properties we show to be sufficient for QCSP tractability were shown in previous work to also guarantee tractability in the context of the CSP. A potentially surprising suggestion of our results is that constraint languages that give rise to tractable subclasses of the CSP tend also to give rise to tractable subclasses of the QCSP!

2 Preliminaries

This section presents the definitions and notation that will be used throughout the paper.

2.1 Tuples, Relations, Constraints, and Formulas

We first wish to define the notion of a *constraint*. Intuitively, a constraint restricts the permissible values of a subset of variables; the permissible values are specified by a relation, while the variable subset is specified by a tuple.

Definition 2.1 *A* relation *of* arity $k \geq 1$ *over set S is a subset of S^k. A* tuple *of arity $k \geq 1$ over set S is an element of S^k. The ith coordinate of a tuple $\overline{x}$ is denoted by $\overline{x}[i]$.*

Definition 2.2 *A* constraint *is a relation γ (over a set A) of arity k paired with a tuple $\overline{x}$ of arity k, and is written $\gamma(\overline{x})$. The set A of the relation is said to be the* domain *of the constraint, and the constraint is said to be over variable set X if $\overline{x}$ is a tuple over X. A constraint $\gamma(\overline{x})$ (with domain A and over variable set X) is* satisfied *by an assignment $f : X \to A$ if $f(\overline{x}) \in \gamma$, where f is applied component-wise to $\overline{x}$. A* constraint language *is a set of relations all of which are over the same domain, and is said to be finite if it contains finitely many relations.*

We next introduce the notion of a quantified formula, a central object of study in this paper; our definition is quite general in that we permit free variables.

Definition 2.3 *A* quantified formula *(over domain A) is an expression of the form*

$$Q_1 v_1 \ldots Q_n v_n \, \mathcal{C}(w_1, \ldots, w_m, v_1, \ldots, v_n)$$

where Q_i is a quantifier from the set $\{\forall, \exists\}$ (for all $i = 1, \ldots, n$), and where $\mathcal{C}$ is a finite set of constraints, each of which is over the domain A and has variables from the set $\{w_1, \ldots, w_m, v_1, \ldots, v_n\}$. The variables $\{w_1, \ldots, w_m\}$ are said to be the free variables *of the formula. If $Q_i = \forall$, then v_i is said to be a $\forall$-variable; if $Q_i = \exists$, then v_i is said to be a $\exists$-variable. When Γ is a constraint language, the formula is said to be over Γ if every constraint in $\mathcal{C}$ has relation from Γ.*[1]

[1]Throughout this paper, we assume that all domains are finite and of nontrivial size (that is, of size greater than or equal to two), and let A denote the domain underlying our relations and formulas.

Truth of a quantified formula (relative to an assignment to the free variables) is defined just as in first-order logic; the set $\mathcal{C}$ is interpreted as the conjunction of the constraints it contains.

Definition 2.4 *Let Γ be a finite constraint language over a finite domain A. The QCSP(Γ) decision problem is to decide, given as input a quantified formula ϕ over Γ without free variables, whether or not ϕ is true.*

The perhaps more familiar *constraint satisfaction problem* is the restricted version of the quantified constraint satisfaction problem where all of the quantifiers must be existential.

Definition 2.5 *Let Γ be a finite constraint language over a finite domain A. The CSP(Γ) decision problem is to decide, given as input an existentially quantified formula ϕ over Γ without free variables, whether or not ϕ is true.*

2.2 Closure Properties

We now define the crucial notion of the *closure properties* of a constraint language, which will give a dual perspective relevant to considerations of complexity. We first fix some notation.

Definition 2.6 *When A is a set and $k \geq 1$ is a natural number, a mapping $f : A^k \to A$ is called an* operation on *A of* rank *k. When A is a set, let O_A denote the set of all operations on A of finite rank; and, let R_A denote the set of all relations over A of finite arity.*

Next, we define how an arbitrary function can be extended to tuples.

Definition 2.7 *For a function $f : S^m \to T$ and tuples $\overline{x_1}, \ldots, \overline{x_m}$ of arity k, $f(\overline{x_1}, \ldots, \overline{x_m})$ is defined to be the arity k tuple*

$$(f(\overline{x_1}[1], \ldots, \overline{x_m}[1]), \ldots, f(\overline{x_1}[k], \ldots, \overline{x_m}[k])).$$

In other words, a function $f : S^m \to T$ acts on m tuples $\overline{x_1}, \ldots, \overline{x_m}$ of equal length coordinate-wise: the ith coordinate of the resulting tuple is equal to $f(\overline{x_1}[i], \ldots, \overline{x_m}[i])$. This action naturally gives a way to define the closure of a relation under a function.

Definition 2.8 *A relation $\gamma \in R_A$ of arity k is* closed under *a rank m operation $f \in O_A$ if for all tuples $\overline{x_1}, \ldots, \overline{x_m} \in \gamma$, it is the case that $f(\overline{x_1}, \ldots, \overline{x_m}) \in \gamma$.*

In turn, we can define the following operators connecting sets of functions to sets of relations.

Definition 2.9 *For a set of operations $F \subseteq O_A$, define*

$$\mathit{Inv}(F) = \{\gamma \in R_A : \gamma \text{ is closed under every function in } F\}.$$

For a constraint language $\Gamma \subseteq R_A$, define

$$\mathit{Pol}(\Gamma) = \{f \in O_A : \text{ every relation in } \Gamma \text{ is closed under } f\}.$$

The Inv and Pol operators form a Galois connection [19].

We now discuss how the closure properties of Γ (that is, Pol(Γ)) are connected to the complexity of Γ.

Definition 2.10 *A quantified formula ϕ with free variable set $\{x_1, \ldots, x_n\}$ is said to induce the relation $\{f(\overline{x}) : \phi$ is satisfied by $f\}$, where $\overline{x} = (x_1, \ldots, x_n)$. When Γ is a constraint language over domain A, let $\langle \Gamma \rangle$ denote the constraint language containing all relations induced by a quantified formula over $\Gamma \cup \{=_A\}$ having only existential quantifiers; and, let $\langle \Gamma \rangle^{\forall}$ denote the constraint language containing all relations induced by a quantified formula over $\Gamma \cup \{=_A\}$. (By $=_A$, we denote the equality relation for the set A, that is, $\{(a, a) : a \in A\}$.)*

We have the following important theorem of Jeavons connecting the relations "expressible" by a constraint language Γ (that is, $\langle \Gamma \rangle$) to the closure properties of Γ.

Theorem 2.11 *[17] For any constraint language $\Gamma \subseteq R_A$, $\langle \Gamma \rangle = Inv(Pol(\Gamma))$.*

From this theorem, we immediately obtain the following proposition.

Proposition 2.12 *[17] For any finite constraint languages $\Gamma_1, \Gamma_2 \subseteq R_A$:*
If $Pol(\Gamma_1) \subseteq Pol(\Gamma_2)$, then $CSP(\Gamma_2) \leq_m^p CSP(\Gamma_1)$.
If $Pol(\Gamma_1) = Pol(\Gamma_2)$, then $CSP(\Gamma_2) \equiv_m^p CSP(\Gamma_1)$.

From Proposition 2.12 we can see that sets of operations of the form $\text{Pol}(\Gamma)$ give us a dual point of view on questions concerning the "complexity" of Γ: if Γ_1 and Γ_2 have the same closure properties, then Γ_1 is tractable if and only if Γ_2 is. That is, the complexity of Γ depends *only* on $\text{Pol}(\Gamma)$! Thus, we can grade the complexity of sets of the form $\text{Pol}(\Gamma)$ according to the complexity of the constraint languages under which such sets are closed. The program of classifying the complexity of constraint languages can consequently be reformulated as a classification question concerning sets of functions $\text{Pol}(\Gamma)$. Fortunately, such sets $\text{Pol}(\Gamma)$ have a particular structure: they are *clones*. We refer the reader to [24, 19, 17, 6, 7] for more information on this dual viewpoint.

Proposition 2.12 also holds in the case of the QCSP (in fact, by essentially the same proof); however, in this case, a stronger statement can be made.

Proposition 2.13 *For any finite constraint languages $\Gamma_1, \Gamma_2 \subseteq R_A$:*
If $Pol(\langle \Gamma_1 \rangle^{\forall}) \subseteq Pol(\Gamma_2)$, then $QCSP(\Gamma_2) \leq_m^p QCSP(\Gamma_1)$.
If $Pol(\langle \Gamma_1 \rangle^{\forall}) = Pol(\langle \Gamma_2 \rangle^{\forall})$, then $QCSP(\Gamma_2) \equiv_m^p QCSP(\Gamma_1)$.

Proposition 2.13 is stronger than Proposition 2.12 (restated to address the QCSP) in light of the following proposition, which shows that the hypothesis $\text{Pol}(\langle \Gamma_1 \rangle^{\forall}) \subseteq \text{Pol}(\Gamma_2)$ implies the hypothesis $\text{Pol}(\Gamma_1) \subseteq \text{Pol}(\Gamma_2)$. For a set of functions $F \subseteq O_A$, let $\text{Surj}(F)$ denote those functions in F that are surjective.

Proposition 2.14 *For all constraint languages Γ, the following containments hold:*
$Surj(Pol(\Gamma)) \subseteq Pol(\langle \Gamma \rangle^{\forall}) \subseteq Pol(\Gamma)$.

We note that the second containment claimed in Proposition 2.14 can be proper.

2.3 Operations on Constraints

The following are two standard operations on constraints.

Definition 2.15 *When $R(\overline{v})$ is a constraint and $\overline{w}$ is a tuple of variables all of which appear in $\overline{v}$, the* projection *of $R(\overline{v})$ onto $\overline{w}$, denoted by $\pi_{\overline{w}}(R(\overline{v}))$, is the constraint $S(\overline{w})$ where*

$$S = \{\overline{a} : \text{ exists assignment } f \text{ satisfying } R(\overline{v}) \text{ such that } f(\overline{w}) = \overline{a}\}.$$

Definition 2.16 *The* join *of two constraints $R_1(\overline{v_1})$ and $R_2(\overline{v_2})$, denoted by $R_1(\overline{v_1}) \bowtie R_2(\overline{v_2})$, is the constraint $R(\overline{v})$, where $\overline{v}$ is the tuple obtained by concatenating the tuples $\overline{v_1}$ and $\overline{v_2}$ and removing variables occurring more than once, and*

$$R = \{\overline{a} : \text{ the assignment } f \text{ such that } f(\overline{v}) = \overline{a} \text{ satisfies } R_1(\overline{v_1}) \text{ and } R_2(\overline{v_2})\}.$$

3 Complexity

In this section, we give a description of the tractable QCSP subclasses (for two-element domains) in terms of Schaefer's description of the tractable CSP subclasses (for two-element domains). The dichotomy theorem on two-element QCSP [10, 11] gives an explicit description of the "QCSP-tractable" constraint languages Γ – those Γ such that QCSP(Γ) is in P. In other words, the boundary between tractability and intractability is precisely delineated. One way to describe the "QCSP-tractable" Γ in domain size two is as exactly those Γ such that Pol(Γ) contains (at least) one of the following four functions: $\wedge$ (the binary function giving the *logical AND*), $\vee$ (the binary function giving the *logical OR*), Δ (the ternary function returning the majority of its arguments), and the ternary function $f(x, y, z) = x+y+z$ (where $+$ is the *logical XOR*). This description can be obtained from [10, Theorem 6.12] along with the characterization lemmas in [10, Chapter 4].

Our theorem in this section links the complexity of the quantified CSP to that of the CSP. Naively, one might conjecture QCSP(Γ) is tractable if and only if CSP(Γ) is tractable – or, in other words, that the boundary of tractability for quantified CSPs is exactly the same as the presumed boundary for CSPs. However, as the following example shows, this naive conjecture fails, even in domain size two: there is a single relation R over $\{0, 1\}$ such that CSP($\{R\}$) is tractable, yet QCSP($\{R\}$) is intractable.

Example 3.1 *Let R be the relation $\{0,1\}^4 \setminus \{(0,0,0,1),(1,1,1,0)\}$, and let N be the relation $\{0,1\}^3 \setminus \{(0,0,0),(1,1,1)\}$. (Constraining three variables using the relation N in effect mandates that their values be "not-all-equal"; the problem CSP($\{N\}$) is sometimes referred to as "not-all-equal SAT.") The problem CSP($\{R\}$) is trivially tractable, since a set of constraints all of which have underlying relation R is satisfiable by the assignment mapping all variables to 0.*

It is straightforward to verify that $\{N\}$ is closed under none of the four functions sufficient for tractability described above, and thus QCSP($\{N\}$) is PSPACE-complete. The relation N can be obtained from R by "universally quantifying away" the last coordinate; more formally, N is in $\langle\{R\}\rangle^{\forall}$. Thus, QCSP($\{N\}$) $\leq_m^p$ QCSP($\{R\}$), and we conclude that QCSP($\{R\}$) is PSPACE-complete.

Before we can link the "QCSP-tractable" constraint languages to the "CSP-tractable" constraint languages given by Schaefer's theorem, we need to define a notion of complexity for QCSP(Γ) when Γ is an *infinte* constraint language.

Definition 3.2 *Let Γ be an infinite constraint language over a finite domain A. We say that CSP(Γ) is tractable if for all finite subsets $\Gamma' \subseteq \Gamma$, CSP(Γ') is in P. Likewise, we say that QCSP(Γ) is tractable if for all finite subsets $\Gamma' \subseteq \Gamma$, QCSP(Γ') is in P. We say that CSP(Γ) (QCSP(Γ)) is intractable if it is not tractable.*

This definition is natural in light of the fact that CSP(Γ) is tractable if and only if CSP($\langle\Gamma\rangle$) is. In addition, QCSP(Γ) is tractable if and only if QCSP($\langle\Gamma\rangle^{\forall}$) is.[2]

An obvious necessary condition for QCSP(Γ) to be tractable is that CSP(Γ) is also tractable, since CSP(Γ) is a particular case of QCSP(Γ). As we discussed, this necessary condition is not sufficient. However, one can make use of Definition 3.2 to formulate a more refined necessary condition for QCSP(Γ) to be

[2]This is because for any finite subset Γ' of $\langle\Gamma\rangle^{\forall}$, QCSP($\Gamma'$) $\leq_m^p$ QCSP(Γ). Likewise, for any finite subset Γ' of $\langle\Gamma\rangle$, CSP(Γ') $\leq_m^p$ CSP(Γ).

tractable: that CSP($\langle\Gamma\rangle^{\forall}$) is tractable. (The tractability of QCSP(Γ) implies the tractability of QCSP($\langle\Gamma\rangle^{\forall}$), implying in turn the tractability of CSP($\langle\Gamma\rangle^{\forall}$).) Interestingly, this necessary condition is sufficient in size two domains.

Theorem 3.3 *For all finite constraint languages Γ over a two-element domain, QCSP(Γ) is tractable if and only if CSP($\langle\Gamma\rangle^{\forall}$) is tractable.*

It is natural to conjecture that Theorem 3.3 extends to larger size domains, that is, for all Γ over a finite domain, the necessary condition (for the tractability of QCSP(Γ)) that CSP($\langle\Gamma\rangle^{\forall}$) is tractable is also sufficient. Suppose that f is an idempotent operation which guarantees that CSP(Γ) is tractable for all Γ closed under f. One prediction of this conjecture is that closure of Γ under any such f also guarantees that QCSP(Γ) is tractable.[3] In Section 5, we will give further evidence for this conjecture by showing that three classes of idempotent operations which have been shown to guarantee tractability of CSP(Γ) (in previous work) also imply the tractability of QCSP(Γ).

4 Proof Systems for QCSPs

This section gives three *proof systems* for QCSPs, which are mechanisms for deriving "implied" constraints from a quantified formula. While all three proof systems are sound, only the first is complete for all quantified formulas; the third is complete for quantified formulas closed under a semilattice operation, and the second is complete for an intermediate class of formulas.

4.1 QCSP-resolution

Definition 4.1 *The constraints derivable by QCSP-resolution from a quantified formula ϕ (without free variables) are those that can be obtained by applying the following rules.*

0. *For all $C \in \phi$, $\phi \vdash C$.*
1. *If $\phi \vdash C$ and C' is a projection of C, then $\phi \vdash C'$.*
2. *If $\phi \vdash R(\overline{v})$ and y is a $\forall$-variable in $\overline{v}$ coming after all other variables (in the quantification order of ϕ) in $\overline{v}$, then $\phi \vdash S(\overline{w})$, where $\overline{w}$ is the tuple $\overline{v}$ with all instances of y removed, and*

   $$S = \{\overline{a} : \forall b \in A, \text{ the extension } f[y = b] \text{ of the assignment } f \text{ defined by } f(\overline{w}) = \overline{a} \text{ satisfies } R\}.$$

   In the case that y is the only variable of $\overline{v}$, if $R \neq A$, then $\phi \vdash \sqcup$.
3. *If $\phi \vdash C_1$ and $\phi \vdash C_2$, then $\phi \vdash C_1 \bowtie C_2$.*

In addition, we write $\phi \vdash \sqcup$ whenever a constraint with empty relation is derivable, that is, if $\phi \vdash \emptyset(\overline{v})$ for some tuple $\overline{v}$. We call $\sqcup$ the empty constraint*.*

Note that the second rule allows one to derive the empty constraint from any constraint over $\forall$-variables that is not a "tautology" in that the underlying relation does not contain all tuples. We note also that if all of the original constraints of a formula ϕ are closed under a surjective operation f, then all constraints derivable from ϕ are also closed under f.

We have that QCSP-resolution is sound and complete.

Theorem 4.2 *A quantified formula ϕ (without free variables) is unsatisfiable if and only if $\phi \vdash \sqcup$ using QCSP-resolution.*

[3] This is because an idempotent function is surjective, and so (when contained in Pol(Γ)) is contained in Pol($\langle\Gamma\rangle^{\forall}$) by Proposition 2.14.

4.2 QCSP-unit-resolution

Definition 4.3 *A constraint $R(\overline{v})$ of a quantified formula* existentially closed *under an operation $f \in O_A$ if for any assignment f to the $\forall$-variables of $\overline{v}$, the set $\{g(\overline{v}) : g$ is an extension of f satisfying $R(\overline{v})\}$ is closed under f.*

Slightly less formally, a constraint $R(\overline{v})$ is existentially closed under an operation if every possible way of "fixing" the $\forall$-variables of $\overline{v}$ yields a constraint closed under f. Definition 4.3 was inspired by the notion of *extended quantified Horn formula*, introduced in [8].

Definition 4.4 *A constraint $R(\overline{v})$ of a quantified formula is an* existential unit constraint *if $\overline{v}$ contains only one $\exists$-variable which comes after all other variables of $\overline{v}$ in the quantification order of the formula.*

Definition 4.5 *The constraints derivable by QCSP-unit-resolution from a quantified formula ϕ (without free variables) are those that can be obtained by applying rules 0-2 of QCSP-resolution (Definition 4.1) as well as the following modification of rule 3:*

If $\phi \vdash_u C_1$, $\phi \vdash_u C_2$, and C_1 is an existential unit constraint, then $\phi \vdash_u C_1 \bowtie C_2$.

(To differentiate QCSP-unit-resolution from QCSP-resolution, we write $\phi \vdash_u C$ if C is derivable from ϕ using QCSP-unit-resolution.)

Although QCSP-unit-resolution is not complete in general, it is complete for a particular class of quantified formulas: those existentially closed under a *semilattice operation*, which by definition is a binary operation that is associative, commutative, and idempotent. When $\oplus$ is a semilattice operation on A, we will let $\leq_{\oplus}$ denote the partial order where $a \leq_{\oplus} c$ if and only if $a \oplus c = c$.

Theorem 4.6 *Suppose that $\oplus$ is a semilattice operation on the domain A, and that ϕ is a quantified formula (over A and without free variables) all of whose constraints are existentially closed under $\oplus$. The formula ϕ is unsatisfiable if and only if $\phi \vdash_u \sqcup$ using QCSP-unit-resolution.*

4.3 QCSP-literal-resolution

Definition 4.7 *A* downward literal *is an expression of the form $v \leq_{\oplus} d$, where v is a variable and $d \in A$; such a literal is satisfied by a mapping $f : V \to d$ defined on v if $f(v) \leq_{\oplus} d$. An* upward literal *is an expression of the form $v \not\leq_{\oplus} d$, where v is a variable and $d \in A$; such a literal is satisfied by a mapping $f : V \to A$ defined on v if it is not the case that $f(v) \leq_{\oplus} d$. A literal is an $\exists$-literal ($\forall$-literal) if its variable is a $\exists$-variable ($\forall$-variable).*

Definition 4.8 *A* Horn-like clause *is a set with downward and upward literals as elements which contains at most one downward literal. A Horn-like clause H over the variable set V is said to be satisfied by an assignment $f : V \to A$ if f satisfies one or more literals in H. When H is a Horn-like clause appearing within a quantified formula, we let $H^{\forall}$ denote the set containing all $\forall$-literals in H; and, we let $H^{\exists}$ denote the set containing all $\exists$-literals in H. A Horn-like clause H appearing within a quantified formula ϕ is an* existential unit clause *if it contains only one $\exists$-literal, the single $\exists$-literal is downward, and for every $\forall$-variable y in H, y comes before the variable of the $\exists$-literal in the quantification order of the formula ϕ.*

Theorem 4.9 *If C is a constraint with relation closed under a semilattice operation $\oplus$, then C is logically equivalent to (that is, has the same satisfying assignments as) the conjunction of Horn-like clauses.*

Theorem 4.9 generalizes Horn's classic theorem that a constraint with relation closed under logical AND ($\wedge$) over the set $\{0, 1\}$ is logically equivalent to the conjunction of Horn clauses [16]. To see this, we let $\oplus = \wedge$; then the only non-trivial downward literal is one of the form $v \leq_\oplus 1$, which is equivalent to the positive literal v; and the only non-trivial upward literal is one of the form $v \not\leq_\oplus 1$, which is equivalent to the negative literal $\overline{v}$.

The third of our proof systems is defined only for quantified formulas closed under a semilattice operation; in fact, we define it directly on formulas with constraint set consisting of Horn-like clauses.

Definition 4.10 *Let ϕ be a quantified formula without free variables and with constraint set consisting only of Horn-like clauses (with respect to the semilattice operation $\oplus$). The Horn-like clauses derivable by QCSP-literal-resolution from the formula ϕ are those that can be obtained by applying the following rules.*

0. *For all $H \in \phi$, $\phi \vdash_l H$.*

1. *If $\phi \vdash_l H_1$, $\phi \vdash_l H_2$, and there exist an $\exists$-variable x and elements $a, b \in A$ such that $(x \leq_\oplus a) \in H_1$ and $(x \leq_\oplus b) \in H_2$, then*
$$\phi \vdash_l (H_1 \setminus \{x \leq_\oplus a\}) \cup (H_2 \setminus \{x \leq_\oplus b\}) \cup L$$
where $L = \{x \leq_\oplus (\oplus\{d \in A : d \leq_\oplus a, d \leq_\oplus b\})\}$ if $\{d \in A : d \leq_\oplus a, d \leq_\oplus b\}$ is non-empty, and $L = \emptyset$ otherwise.

2. *If $\phi \vdash_l H$ and there exist $a \in A$ and an $\exists$-variable x such that $(x \leq_\oplus a) \in H$, then*
$$\phi \vdash_l (H \setminus \{x \leq_\oplus a\}) \cup \{x \leq_\oplus b\}$$
for all $b \in A$ such that $a \leq_\oplus b$.

3. *If $\phi \vdash_l U$, $\phi \vdash_l H$, U is an existential unit clause with downward literal $(x \leq_\oplus a)$, and $(x \not\leq_\oplus a) \in H$, then*
$$\phi \vdash_l (U \setminus \{x \leq_\oplus a\}) \cup (H \setminus \{x \not\leq_\oplus a\}).$$

4. *If $\phi \vdash_l H$, y is a $\forall$-variable which is the last variable in the quantification order of ϕ occurring in H, and there exists a value $a \in A$ such that the assignment $y = a$ does not satisfy H_y (the clause containing all y-literals in H), then*
$$\phi \vdash_l H \setminus H_y.$$

Theorem 4.11 *Let ϕ be a quantified formula without free variables and with constraint set consisting only of Horn-like clauses (with respect to the semilattice operation $\oplus$). The formula ϕ is unsatisfiable if and only if $\phi \vdash_l \emptyset$ using QCSP-literal-resolution.*

5 Algorithms

In this section, we present our tractability results. As we will note, all of the sufficient conditions we give for QCSP tractability were previously shown to imply CSP tractability.

Semilattice operations were shown to guarantee CSP tractability in [24]. Every semilattice operation $\oplus$ has a *top element* $\mathbf{t}$ with the property that $a \leq_\oplus \mathbf{t}$ for all $a \in A$. If a semilattice operation has a *bottom element* – that is, an element $\mathbf{b}$ such that $\mathbf{b} \leq_\oplus a$ for all $a \in A$ – we say that it is a semilattice operation *with bottom*.

Theorem 5.1 *If Γ is a finite constraint language (over a finite domain) closed under a semilattice operation with bottom, then QCSP(Γ) is in P.*

The proof of Theorem 5.1 makes crucial use of the QCSP-literal-resolution proof system given in the previous section.

Feder and Vardi [14] showed the tractability of "subgroup problems" or, equivalently, the CSP with constraint language closed under a *coset operation*.

Definition 5.2 *A ternary operation $f : G^3 \rightarrow G$ is a* coset operation *if it is of the form $f(a, b, c) = a * (b^{-1}) * c$, where $*$ is a binary operation and $^{-1}$ is a unary operation such that $(G, *, ^{-1})$ is a group.*

Theorem 5.3 *If Γ is a finite constraint language (over a finite domain) closed under a coset operation, then QCSP(Γ) is in P.*

Multiple characterizations of constraint languages closed under *near-unanimity operations* were given in [18], where it was shown that such constraint languages give rise to a tractable CSP subclass.

Definition 5.4 *An operation $f : A^k \rightarrow A$ of rank $k \geq 3$ is said to be a* near-unanimity *operation if the following property holds: for all $i \in \{1, \ldots, k\}$ and $\bar{t} \in A^k$, if there exists $a \in A$ such that $\bar{t}[j] = a$ for all $j \in \{1, \ldots, k\} \setminus \{i\}$, then $f(\bar{t}) = a$.*

In other words, f is near-unanimity if for all $a, b \in A$, the following equations hold: $f(a, b, b, \ldots, b, b, b) = f(b, a, b, \ldots, b, b, b) = f(b, b, b, \ldots, b, a, b) = f(b, b, b, \ldots, b, b, a)$.

Theorem 5.5 *If Γ is a finite constraint language closed under a near-unanimity operation, then QCSP(Γ) is in P.*

The three sufficient conditions for tractability given in this section strictly generalize all of the tractability results in domain size two. Recall from Section 3 that the "tractable" Γ (in domain size two) are exactly those Γ such that Pol(Γ) contains one of four functions: $\wedge$, $\vee$, Δ, and $f(x, y, z) = x + y + z$ (described in Section 3). It is easily verified that $\wedge$ and $\vee$ are semilattice operations with bottom, $f(x, y, z)$ is a coset operation, and Δ is a near-unanimity operation.

6 Future Work

We believe that the results obtained in this paper attest to the power of the dual algebraic viewpoint used to study the CSP in previous work. An obvious open question is to classify the complexity of constraint languages that have not been studied in this paper. The theorem and conjecture given in Section 3 may give clues as to how one might proceed. Our results seem to indicate that restrictions on the constraint language giving rise to tractable CSP subclasses also give rise to tractable QCSP subclasses. It would be of great interest to continue investigating the extent to which this phenomenon persists. More broadly, it would be quite intriguing if *other* types of restrictions yielding tractable CSP subclasses also induce tractable QCSP subclasses, and we consider studying other restrictions to be an important direction for future work.

Acknowledgements. The author wishes to thank Dexter Kozen for useful discussions and comments.

References

[1] Bengt Aspvall, Michael F. Plass, and Robert Endre Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 8(3):121–123, 1979.

[2] E. Böhler and H. Vollmer. Boolean functions and post's lattice with applications to complexity theory. Lecture Notes for Logic and Interaction 2002.

[3] Andrei Bulatov and Peter Jeavons. Tractable constraints closed under a binary operation. Technical Report PRG-TR-12-00, Oxford University, 2000.

[4] Andrei A. Bulatov. A dichotomy theorem for constraints on a three-element set. In *Proceedings of 43rd IEEE Symposium on Foundations of Computer Science*, pages 649–658, 2002.

[5] Andrei A. Bulatov. Malt'sev constraints are tractable. Technical Report PRG-RR-02-05, Oxford University, 2002.

[6] Andrei A. Bulatov, Andrei A. Krokhin, and Peter Jeavons. Constraint satisfaction problems and finite algebras. In *Proceedings 27th International Colloquium on Automata, Languages, and Programming – ICALP'00*, volume 1853 of *Lecture Notes In Computer Science*, pages 272–282, 2000.

[7] Andrei A. Bulatov, Andrei A. Krokhin, and Peter Jeavons. The complexity of maximal constraint languages. In *ACM Symposium on Theory of Computing*, pages 667–674, 2001.

[8] Hans Kleine Büning, Marek Karpinski, and Andreas Flögel. Resolution for quantified boolean formulas. *Information and Computation*, 117(1):12–18, 1995.

[9] Nadia Creignou and Miki Hermann. Complexity of generalized satisfiability counting problems. *Information and Computation*, 125(1):1–12, 1996.

[10] Nadia Creignou, Sanjeev Khanna, and Madhu Sudan. *Complexity Classification of Boolean Constraint Satisfaction Problems*. SIAM Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, 2001.

[11] Victor Dalmau. Some dichotomy theorems on constant-free quantified boolean formulas. Technical Report LSI-97-43-R, Llenguatges i Sistemes Informàtics - Universitat Politècnica de Catalunya, 1997.

[12] Victor Dalmau. A new tractable class of constraint satisfaction problems. In *6th International Symposium on Artificial Intelligence and Mathematics*, 2000.

[13] Victor Dalmau, Phokion G. Kolaitis, and Moshe Y. Vardi. Constraint satisfaction, bounded treewidth, and finite-variable logics. In *Constraint Programming '02*, Lecture Notes In Computer Science, 2002.

[14] Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic snp and constraint satisfaction: A study through datalog and group theory. *SIAM J. Comput.*, 28(1):57–104, 1998.

[15] M. Furst, J. E. Hopcroft, and E. Luks. Polynomial time algorithms for permutation groups. In *Proc. 21st IEEE Symp. on Foundations of Comp. Sci.*, pages 36–41, 1980.

[16] A. Horn. On sentences which are true of direct unions of algebras. *Journal of Symbolic Logic*, 16:14–21, 1951.

[17] Peter Jeavons. On the algebraic structure of combinatorial problems. *Theoretical Computer Science*, 200:185–204, 1998.

[18] Peter Jeavons, David Cohen, and Martin Cooper. Constraints, consistency, and closure. *Articial Intelligence*, 101(1-2):251–265, 1998.

[19] Peter Jeavons, David Cohen, and Justin Pearson. Constraints and universal algebra. *Annals of Mathematics and Artificial Intelligence*, 24(1-4):51–67, 1998.

[20] D. Kavvadias and M. Sideri. The inverse satisfiability problem. *SIAM Journal on Computing*, 28(1):152–163, 1998.

[21] Sanjeev Khanna, Madhu Sudan, Luca Trevisan, and David P. Williamson. The approximability of constraint satisfaction problems. *SIAM Journal on Computing*, 30(6):1863–1920, 2001.

[22] Phokion G. Kolaitis and Moshe Y. Vardi. A game-theoretic approach to constraint satisfaction. In *AAAI/IAAI*, pages 175–181, 2000.

[23] Justin K. Pearson and Peter G. Jeavons. A survey of tractable constraint satisfaction problems. Technical report, Royal Holloway, University of London, 1997.

[24] P.G.Jeavons, D.A.Cohen, and M.Gyssens. Closure properties of constraints. *Journal of the ACM*, 44:527–548, 1997.

[25] Emil L. Post. *The Two-Valued Iterative Systems of Mathematical Logic*. Princeton University Press, 1941.

[26] Steffen Reith and Heribert Vollmer. Optimal satisfiability for propositional calculi and constraint satisfaction problems. In *Proc. 25th International Symposium on Mathematical Foundations of Computer Science*, volume 1893 of *Springer Lecture Notes in Computer Science*, pages 640–649, 2000.

[27] Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 216–226, 1978.

A Proofs

A.1 Proof of Theorem 3.3

Proof. We break into two cases depending on whether or not $\mathrm{Pol}(\langle\Gamma\rangle^{\forall})$ contains one of the four operations sufficient for tractability described above. If so, then $\mathrm{Pol}(\Gamma)$ also contains one of the four operations, by Proposition 2.14, since each of the four operations is surjective. Thus QCSP(Γ) is tractable; in addition, CSP($\langle\Gamma\rangle^{\forall}$) is also tractable by Schaefer's dichotomy theorem.

If not, then by Post's classification theorem of closed functions [25, 2], $\mathrm{Pol}(\langle\Gamma\rangle^{\forall})$ is contained in the clone generated by the following three functions: logical NOT (a unary function), the constant function 0, and the constant function 1. It follows that the relation R given in Example 3.1 is in $\langle\Gamma\rangle^{\forall}$ (as R is closed under these three functions); since QCSP($\{R\}$) is PSPACE-complete, QCSP(Γ) is also PSPACE-complete. In addition, CSP($\{R\}$) is NP-complete by Schaefer's dichotomy theorem. $\Box$

A.2 Proof of Theorem 4.2

Before proving Theorem 4.2, we prove the following lemma. When $\phi = Q_1 v_1 \ldots Q_n v_n \mathcal{C}(v_1, \ldots, v_n)$ is a quantified formula and $a \in A$, we let $\phi[v_1 = a]$ denote the quantified formula $Q_2 v_2 \ldots Q_n v_n \mathcal{C}'(v_2, \ldots, v_n)$, where $\mathcal{C}' = \{C[v_1 = a] : C \in \mathcal{C}\}$ and the constraint $C[v_1 = a]$ is obtained by joining C with the constraint $\{a\}(v_1)$ and then projecting onto the original variable tuple of C omitting v_1.

Lemma A.1 *Suppose $\phi = Q_1 v_1 \ldots Q_n v_n \mathcal{C}(v_1, \ldots, v_n)$ is a quantified formula (without free variables), and that $a \in A$. If $\phi[v_1 = a] \vdash \sqcup$, then $\phi \vdash S(v_1)$ for some set $S \subseteq A$ not containing a.*

Proof. Suppose that $C_1, \ldots, C_m = \sqcup$ is a QCSP-resolution derivation of $\sqcup$ from $\phi[v_1 = a]$. By examining the QCSP-resolution derivation rules, it can be shown by induction on on m that there is a QCSP-resolution derivation $C'_1, \ldots, C'_m$ from ϕ such that for all $i = 1, \ldots, m$, $C'_i[v_1 = a] = C_i$. Thus, C'_m is a constraint satisfying the desired property. $\Box$

Proof of Theorem 4.2. Let $\phi = Q_1 v_1 \ldots Q_n v_n \mathcal{C}$ be a quantified formula (without free variables). It is straightforward to show that for any set of constraints $\mathcal{D}$ such that $C \in \mathcal{D}$ implies $\phi \vdash C$, ϕ is satisfiable if and only if $Q_1 v_1 \ldots Q_n v_n \mathcal{C} \cup \mathcal{D}$ is satisfiable. This gives the "if" direction.

Now assume that ϕ is unsatisfiable. We prove that $\phi \vdash \sqcup$, by induction on the number of quantifiers n.

In the base case, $n = 1$. If $Q_1 = \exists$, then the join of all constraints in $\mathcal{C}$ is empty (since ϕ is unsatisfiable), and thus $\sqcup$ is derivable from ϕ by repeated applications of QCSP-resolution rule 3. If $Q_1 = \forall$, then the unsatisfiability of ϕ implies that the relation of some constraint C is not equal to A. By applying QCSP-resolution rule 2 to C, the empty constraint $\sqcup$ is derived.

Suppose that $n > 1$. If $Q_1 = \exists$, then by induction $\phi[v_1 = a] \vdash \sqcup$ for all $a \in A$. By Lemma A.1, for each $a \in A$ there exists a set $S_a \subseteq A$ such that $\phi \vdash S_a(v_1)$ and $a \notin A$. Joining all such constraints $S_a(v_1)$ using rule 3 yields the empty constraint $\sqcup$. If $Q_1 = \forall$, then there exists $a \in A$ such that $\phi[v_1 = a]$ is unsatisfiable, and thus by induction $\phi[v_1 = a] \vdash \sqcup$. By Lemma A.1, $\phi \vdash S(v_1)$ for some set $S \subseteq A$ not containing a. By applying rule 2 to $S(v_1)$, $\phi \vdash \sqcup$. $\Box$

A.3 Proof of Theorem 4.6

Before proving this theorem, we give a useful characterization of satisfiable quantified formulas. Define a *strategy* for such a formula of the form $\forall y_1 \exists x_1 \ldots \forall y_n \exists x_n \mathcal{C}(y_1, x_1, \ldots, y_n, x_n)$ to be a sequence of mappings $\tau_1 : A^1 \to A, \ldots, \tau_n : A^n \to A$. We say that a strategy is a *winning strategy* if for all mappings $f : \{y_1, \ldots, y_n\} \to A$, the assignment τ_f mapping x_i to $\tau_i(f(y_1), \ldots, f(y_i))$ and y_i to $f(y_i)$ (for all

$i = 1, \ldots, n$) satisfies every constraint in $\mathcal{C}$. It can be verified that a quantified formula is satisfiable if and only if there exists a winning strategy for the formula.

Proof. The "if" direction follows from Theorem 4.2, since QCSP-unit-resolution is a restricted version of QCSP-resolution. We thus prove the "only if" direction. Assume without loss of generality that ϕ has the form $\forall y_1 \exists x_1 \ldots \forall y_n \exists x_n \mathcal{C}(y_1, x_1, \ldots, y_n, x_n)$, and suppose that it is not the case that $\phi \vdash_u \sqcup$. Let $f_{\{a_i : i=1,\ldots,k\}}$ denote the mapping taking y_i to a_i for all $i = 1, \ldots, k$. Define (for $k = 1, \ldots, n$) $\tau_k(a_1, \ldots, a_k)$ to be

$$\oplus\{a : f_{\{a_i:i=1,\ldots,k\}}[x_k = a] \text{ satisfies all existential unit constraints } C \text{ containing } x_k \text{ such that } \phi \vdash_u C\}.$$

Note that the set in this definition is non-empty, and so τ_k is well-defined. (If the set were empty for some k, joining all derivable existential unit constraints C containing x_k would yield a constraint C' with only $\forall$-variables not satisfiable by $f_{\{a_i:i=1,\ldots,k\}}$; then, from C', the empty constraint $\sqcup$ would be derivable by rule 2.)

We show that the τ_k form a winning strategy. Let $f : \{y_1, \ldots, y_n\} \to A$ be an assignment to the $\forall$-variables; we wish to show that τ_f satisfies all constraints in $\mathcal{C}$. It suffices to show that τ_f satisfies all C such that $\phi \vdash_u C$. We prove this by induction on the number t of $\exists$-variables that C contains.

We split into three cases.

Case 1: $t = 0$. If τ_f does not satisfy C, then the empty constraint $\sqcup$ can be derived from C using rule 2.

Case 2: $t = 1$. Let x denote the $\exists$-variable in C. Obtain C' by applying rule 2 to C until x comes after all other variables in the order of quantification (of ϕ) – that is, until C becomes an existential unit constraint. It suffices to show that C' is satisfied by τ_f; this follows immediately from the definition of the τ_k.

Case 3: $t > 1$. Obtain C' by applying rule 2 to C until the last variable in the order of quantification is an $\exists$-variable; denote this $\exists$-variable by x. For each $\exists$-variable v in C', let U_v denote the constraint equal to the join of all derivable existential unit constraints containing v. Obtain the constraint C'' by joining C' with all of the U_v. It suffices to show that C'' is satisfied by τ_k. Let $\overline{t}$ be a tuple in the relation of C'' such that the coordinate corresponding to variable x, which we denote by $\overline{t}[x]$, is equal to $\tau_f(x)$. By induction, the projection of C'' onto all variables except x is satisfied by τ_f, so there exists a tuple $\overline{p}$ in the relation of C'' equal to τ_f at all coordinate locations except (possibly) x. For all $\forall$-variables y, $\overline{p}[y] = \overline{t}[y]$; for the $\exists$-variable x, $\overline{p}[x] \leq_\oplus \overline{t}[x]$; and for all other $\exists$-variables w, $\overline{t}[w] \leq_\oplus \overline{p}[w]$. (This is because of the definition of the τ_k and the fact that we performed the joins with the U_v.) By the existential closure of C'' under $\oplus$ and the fact that $\overline{p}$ and $\overline{t}$ agree at all $\forall$-variables, we have $\tau_f(\overline{v}) = \overline{p} \oplus \overline{t} \in R$. $\square$

A.4 Proof of Theorem 4.9

Proof. Let $C = R(\overline{v})$ be a constraint where R is closed under a semilattice operation $\oplus$. We assume without loss of generality that $\overline{v}$ contains no variable more than once. It suffices to show that for each $\overline{a} \notin R$, there exists a Horn-like clause $H_{\overline{a}}$ such that any assignment satisfying C satisfies $H_{\overline{a}}$, but the assignment f defined by $f(\overline{v}) = \overline{a}$ does not satisfy $H_{\overline{a}}$. Fix $\overline{a} \notin R$, and define $S_{\overline{a}} = \{\overline{b} : \overline{b} \in R, \overline{b} \leq_\oplus \overline{a}\}$ where $\leq_\oplus$ is extended to a partial ordering on tuples by defining $\overline{s} \leq_\oplus \overline{t}$ if and only if at all coordinates i, $\overline{s}[i] \leq_\oplus \overline{t}[i]$.

If $S_{\overline{a}}$ is empty, then it can be verified that the Horn-like clause $H_{\overline{a}} = \bigcup_{i=1}^{r}\{v_i \not\leq_\oplus a_i\}$ has the desired properties, where r denotes the length of the tuples $\overline{v}$, $\overline{a}$, and $\overline{m}$. Otherwise, define $\overline{m} = \oplus S_{\overline{a}}$; it is straightforward to verify that $S_{\overline{a}}$ is closed under $\oplus$, so $\overline{m} \in S_{\overline{a}}$. Since $S_{\overline{a}} \subseteq R$, $\overline{m} \neq \overline{a}$, and $\overline{m}$ and $\overline{a}$ differ at some coordinate, say coordinate j (that is, $\overline{m}[j] \neq \overline{a}[j]$). It can be verified that the Horn-like clause $H_{\overline{a}} = (\bigcup_{i=1}^{r}\{v_i \not\leq_\oplus a_i\}) \cup \{v_j \leq_\oplus m_j\}$ has the desired properties. $\square$

A.5 Proof of Theorem 4.11

Proof. As with the other resolution proof systems, the "if" direction is straightforward, so we focus on the "only if" direction. Assume without loss of generality that ϕ has the form $\forall y_1 \exists x_1 \ldots \forall y_n \exists x_n \mathcal{C}(y_1, x_1, \ldots, y_n, x_n)$, where $\mathcal{C}$ consists only of Horn-like clauses, and suppose that it is not the case that $\phi \vdash_u \sqcup$. Let $f_{\{a_i : i=1,\ldots,k\}}$ denote the mapping taking y_i to a_i for all $i = 1, \ldots, k$. Define (for $k = 1, \ldots, n$)

$$\tau_k(a_1, \ldots, a_k) = \oplus\{a : f_{\{a_i:i=1,\ldots,k\}}[x_k = a]$$
$$\text{satisfies all existential unit clauses } C \text{ containing } x_k \text{ such that } \phi \vdash_l C\}.$$

We claim that the τ_k form a winning strategy. To prove this, we first show that the τ_k are well-defined, that is, the intersection $\cap\{a : f_{\{a_i:i=1,\ldots,k\}}[x_k = a] \text{ satisfies } C\})$ is always non-empty. We show this by contradiction: suppose that for some k, the intersection is empty. Then, for every $a \in A$ there exists an existential unit clause H_a such that $\phi \vdash_l H_a$ and $f_{\{a_i:i=1,\ldots,k\}}[x_k = a]$ does not satisfy H_a. We have (for all $a \in A$) that $f_{\{a_i:i=1,\ldots,k\}}$ does not satisfy $H_a^{\forall}$, and that $x_k = a$ does not satisfy the single downward literal in $H_a^{\exists}$. Applying rule 1 to all of the H_a, we obtain $\phi \vdash_l \bigcup_{a \in A} H_a^{\forall}$. Since $f_{\{a_i:i=1,\ldots,k\}}$ does not satisfy $\bigcup_{a\in A} H_a^{\forall}$, by repeated applications of rule 4, we obtain that $\phi \vdash_l \emptyset$, a contradiction.

We show that the τ_k form a winning strategy. Let $f : \{y_1, \ldots, y_a\} \to A$ be an assignment to the $\forall$-variables of ϕ; we wish to show that τ_f satisfies all clauses of ϕ. It suffices to show that τ_f satisfies all clauses H such that $\phi \vdash_l H$. We prove this by induction on the number t of upward $\exists$-literals that H contains.

We split into three cases.

Case 1: $t = 0$ and H does not contain any $\exists$-literals. If τ_f does not satisfy H, then the empty set can be derived from H using rule 4.

Case 2: $t = 0$ and H contains a downward $\exists$-literal. Obtain H' by applying rule 4 to eliminate $\forall$-literals in H until rule 4 is no longer applicable. If the last variable in H' (relative to the quantification order of ϕ) is a $\forall$-variable y, then for all $a \in A$, every extension of the assignment $y = a$ satisfies H', and so in particular H' is satisfied by f. If the last variable in H' is an $\exists$-variable x, then H' is an existential unit clause and is satisfied by τ_f by the definition of the τ_k.

Case 3: H contains an upward $\exists$-literal $x_k \not\leq_{\oplus} d$. Suppose (for a contradiction) that τ_f does not satisfy H; then $\tau_f(x_k) \leq d$. By reasoning similar to the above argument that the τ_i were well-defined, there is a derivable existential unit clause U with literal $x_k \leq_{\oplus} \tau_f(x_k)$ such that τ_f does not satisfy $U \setminus \{x_k \leq_{\oplus} \tau_f(x_k)\}$. Set $U' = (U \setminus \{x \leq_{\oplus} \tau_f(x_k)\}) \cup \{x_k \leq d\}$. By applying rule 2 to U, we obtain $\phi \vdash_l U'$. Set $H' = (U' \setminus \{x_k \leq_{\oplus} d\}) \cup (H \setminus \{x_k \not\leq_{\oplus} d\})$. By rule 3, $\phi \vdash_l H'$. Notice that H' is not satisfied by τ_f; this contradicts our inductive hypothesis, since H' contains one less upward $\exists$-literal than H. $\square$

A.6 Proof of Theorem 5.1

Proof. Let $\phi = \forall y_1 \exists x_1 \ldots \forall y_n \exists x_n \mathcal{C}$ be an instance of the QCSP(Γ) problem, and let $\oplus$ be a semilattice operation with bottom element $\mathbf{b}$ and top element $\mathbf{t}$ under which Γ is closed. By appeal to Theorem 4.9, we assume that $\mathcal{C}$ contains Horn-like clauses. We assume without loss of generality that no literals in $\mathcal{C}$ are trivial, that is, of the form $v \leq_{\oplus} \mathbf{t}$ or $v \not\leq_{\oplus} \mathbf{t}$.

Let $D^{\exists}$ denote the subset of $\mathcal{C}$ containing all clauses with a downward $\exists$-literal; Let $D^{\forall}$ denote the subset of $\mathcal{C}$ containing all clauses with a downward $\forall$-literal; and let U denote the subset of $\mathcal{C}$ containing all clauses having only upward literals. It can be verified that every clause derivable from ϕ by QCSP-literal-resolution is derivable from $\phi_H \stackrel{\text{def}}{=} \forall y_1 \exists x_1 \ldots \forall y_n \exists x_n (D^{\exists} \cup \{H\})$ for some $H \in D^{\forall} \cup U$.

Deciding whether or not ϕ is satisfiable thus amounts to deciding whether or not ϕ_H is satisfiable, for all $H \in D^{\forall} \cup U$. We thus show how to decide satisfiability of ϕ_H; we split into two cases.

Case $H \in U$: The formula ϕ_H is satisfiable if and only if the CSP instance

$$\exists x_1 \ldots \exists x_n \{C \setminus C_\forall : C \in D^\exists \cup \{H\}\}$$

is satisfiable (where $C_\forall$ denotes the set of all $\forall$-literals in the clause C). If the CSP instance is satisfiable, a satisfying assignment for it gives a winning strategy for ϕ_H (which ignores the $\forall$-player); if the CSP instance is unsatisfiable, ϕ_H is unsatisfiable as the $\forall$-player can set all $\forall$-variables to $\mathbf{b}$ to falsify all $\forall$-literals, causing the $\exists$-player to lose. Satisfiability of the CSP instance can be decided in polynomial time, using for example the algorithm of [24].

Case $H \in D^\forall$: Let y denote the variable in the downward literal of H. Remove all $\forall$-literals not over y from the clauses of $D^\exists \cup \{H\}$ to obtain the set of clauses $\mathcal{C}'$; ϕ_H is satisfiable if and only if $\phi' = \forall y_1 \exists x_1 \ldots \forall y_n \exists x_n \mathcal{C}'$ is satisfiable, as a QCSP-literal-resolution derivation of $\emptyset$ from ϕ_H gives a derivation of $\emptyset$ from ϕ' by removal of $\forall$-literals not including y; and, a derivation of $\emptyset$ from ϕ' gives a derivation of derivation of $\emptyset$ from ϕ_H by adding in $\forall$-literals as appropriate to derive (from ϕ_H) a clause consisting only of $\forall$-variables, from which $\emptyset$ can be derived by repeated applications of QCSP-literal-resolution rule 4 (with $a = \mathbf{b}$). Notice that when the last variable in a clause in the quantification order is a $\forall$-variable y, all literals involving y can be removed by using rule 4 with $a = \mathbf{b}$.

We now show how to decide if $\phi' = \forall y_1 \exists x_1 \ldots \forall y_n \exists x_n \mathcal{C}'$ is satisfiable. Let y_i denote the single $\forall$-variable occurring in $\mathcal{C}'$. It suffices to compute the values that the winning strategy assigns to the $\exists$-variables $\{x_1, \ldots, x_{i-1}\}$ coming before y, since then ϕ' is satisfiable if and only if ϕ_a is satisfiable for all $a \in A$, where ϕ_a is the existentially quantified formula $\exists x_i \ldots \exists x_n \mathcal{C}'_a$ with $\mathcal{C}'_a$ obtained from $\mathcal{C}'$ by instantiating $\{x_1, \ldots, x_{i-1}\}$ with the values given by the winning strategy, and instantiating y_i with a.

In order to compute the values that the winning strategy assigns to $\{x_1, \ldots, x_{i-1}\}$, we compute the most stringent existential unit clauses over $\{x_1, \ldots, x_{i-1}\}$ derivable from ϕ'; notice that such an existential unit clause contains a single downward literal. (We consider a literal $x \leq_\oplus a$ to be more stringent than another literal $x \leq_\oplus c$ if $a \leq_\oplus c$; a "most stringent" literal exists by rule 2.) Let ϕ'' be the formula ϕ', but with all y_i-literals (and hence all $\forall$-literals) removed. It is easy to see that the existential unit clauses over $\{x_1, \ldots, x_{i-1}\}$ derivable from ϕ'' are exactly those derivable from ϕ'.

We thus compute the most stringent downward literals $\{x_1, \ldots, x_{i-1}\}$ derivable from ϕ'', using the following algorithm:

- Set $a_1, \ldots, a_n$ all equal to $\mathbf{t}$.
- While there exists an unmarked clause H such that the mapping taking x_j to a_j (for all $j = 1, \ldots, n$) satisfies all upward literals in H, do:
  - Terminate and report "unsatisfiable" if H contains no downward literal.
  - Letting $x_k \leq_\oplus a$ be the downward literal of H, update a_k to have the value $\oplus\{d : d \leq_\oplus a, d \leq_\oplus a_k\}$.
  - Mark H.

It is easily verified that at any point during the execution of the algorithm, $x_j \leq_\oplus a_j$ is derivable from ϕ'' using QCSP-literal-resolution. Thus, if the algorithm terminates and reports "unsatisfiable", it does so correctly. Moreover, it can be verified that the map taking x_j to the final value of a_j (following a successful halting of the algorithm) satisfies ϕ'', so the literals $x_j \leq_\oplus a_j$ are the most stringent existential unit clauses derivable, and the winning strategy for ϕ' maps x_j to a_j for all $j = 1, \ldots, i-1$. $\Box$

A.7 Proof of Theorem 5.3

Before proving the theorem, we require the following lemma, which states that a relation is closed under a coset operation if and only if it is a coset of a power of the underlying group.

Lemma A.2 *A relation $R \subseteq G^k$ is closed under a coset operation $f(a,b,c) = a * (b^{-1}) * c$ if and only if R is a coset of the group $(G, *, ^{-1})^k$.*

The proof of this lemma is straightforward, and omitted. We suspect that it has appeared previously in the literature, but do not know of a reference.

Proof. Let $\phi = Q_1 v_1 \dots Q_n v_n \mathcal{C}$ be an instance of the QCSP closed under the coset operation $f(a,b,c) = a * (b^{-1}) * c$ with underlying group $(G, *, ^{-1})^k$. By Lemma A.2, we can view $\mathcal{C}$ as containing s constraints of the form

$$(v_1, \dots, v_n) \in a_i J_i$$

where $a_1, \dots, a_s$ are elements of G^n, and $J_1, \dots, J_s$ are subgroups of G^n. Define $H_0 = G^n$ and $H_i = J_i \cap H_{i-1}$ for $i = 1, \dots, s$. Define $K_0 = H_s$ and $K_i = K_0 \cap \{\bar{t} \in G^n : t[j] = e \text{ for all } j = 1, \dots, i\}$ for $i = 1, \dots, n$. We have the chain of subgroups

$$G^n = H_0 > \cdots > H_s = K_0 > \cdots > K_n = \{(e, \dots, e)\}.$$

Using the algorithm due to Furst et al. [15] we can compute, in polynomial time, coset representatives for this chain of subgroups – that is, subsets $P_1, \dots, P_s$ and $R_1, \dots, R_n$ of G^n such that $H_i = \cup_{p \in P_{i+1}} p H_{i+1}$ (for $i = 0, \dots, s-1$), and $K_i = \cup_{r \in R_{i+1}} r K_{i+1}$ (for $i = 0, \dots, n-1$) where the unions are disjoint.

Next, we compute the set of elements satisfying the CSP $\mathcal{C}$ (that is, we compute the set $\cap_{i=1}^{s} a_i J_i$) as follows. We attempt to find group elements $p_1 \in P_1, \dots, p_s \in P_s$ such that for all $i = 1, \dots, s$,

$$(\prod_{k=1}^{i} p_k) H_i = \bigcap_{k=1}^{i} a_k J_k.$$

We claim that such elements p_i exist unless the right hand side intersection is empty. To compute the p_i, we begin by setting p_1 so that $p_1 H_1 = a_1 J_1$. For $i \geq 2$, define p_i to be the element of p_i such that $p_i \in p_{i-1}^{-1} \dots p_1^{-1} a_i J_i$; if there is no such p_i, the intersection is empty, and the QCSP ϕ is clearly unsatisfiable.

Assume that $p_1, \dots, p_s$ were successfully computed. The given QCSP instance ϕ is equivalent to the QCSP instance

$$\psi = Q_1 v_1 \dots Q_n v_n [(v_1, \dots, v_n) \in (q_1, \dots, q_n) K_0]$$

where $(q_1, \dots, q_n) \stackrel{\text{def}}{=} \prod_{k=1}^{i} p_k$. We simplify a QCSP of this form into another of this form according to the following two rules, which describe how to eliminate the innermost quantifier and variable. (Applying these rules iteratively can be done in polynomial time.)

Case $Q_n = \exists$: ψ is satisfiable if and only if $\psi' = Q_1 v_1 \dots Q_{n-1} v_{n-1} [(v_1, \dots, v_{n-1}) \in (q_1, \dots, q_{n-1}) p(K_0)]$ is satisfiable, where $p : G^n \to G^{n-1}$ is the projection homomorphism onto the first $n-1$ coordinates.

Case $Q_n = \forall$: ψ is satisfiable if and only if ψ' is satisfiable and $|K_{n-1}| = |G|$. When $|K_{n-1}| = |G|$, any setting $(a_1, \dots, a_{n-1})$ to the variables $(v_1, \dots, v_{n-1})$ satisfying $(q_1, \dots, q_{n-1}) p(K_0)$ has the property that every extension satisfies $(q_1, \dots, q_n) K_0$, that is, for all $g \in G$, $(a_1, \dots, a_{n-1}, g) \in (q_1, \dots, q_n) K_0$. When $|K_{n-1}| < |G|$, for any setting $(a_1, \dots, a_{n-1})$ to the variables $(v_1, \dots, v_{n-1})$, there exists an element $g \in G$ such that $(a_1, \dots, a_{n-1}, g) \notin (q_1, \dots, q_n) K_0$. $\square$

A.8 Proof of Theorem 5.5

Before proving this theorem, we introduce notions of *consistency*.

Definition A.3 *Let $\mathcal{C}$ be a set of constraints over variable set V. For a subset V' of V, a mapping $f : V' \to A$ is a* partial solution *to $\mathcal{C}$ if for every constraint $R(\overline{v}) \in \mathcal{C}$, there is an extension $f' : V' \cup \overline{v} \to A$ of f satisfying $R(\overline{v})$.*[4] *The set $\mathcal{C}$ is j-*consistent *if for every size $j-1$ subset V' of V and for every variable $v \in V \setminus V'$, any partial solution $f : V' \to A$ of $\mathcal{C}$ can be extended to a partial solution $f' : V' \cup \{v\} \to A$ of $\mathcal{C}$. The set $\mathcal{C}$ is* strongly k-consistent *if it is j-consistent for $j = 2, \ldots, k$. The set $\mathcal{C}$ is* globally consistent *if it is strongly $|V|$-consistent.*

The following theorem shows that when a constraint language is closed under a near-unanimity operation, ensuring a sufficiently high (but constant) degree of "local" consistency implies global consistency.

Theorem A.4 *[18] Suppose that Γ is a constraint language closed under a near-unanimity operation of rank r. If a set of constraints $\mathcal{C}$ over Γ is strongly r-consistent, then $\mathcal{C}$ is globally consistent.*

We now turn to the proof of Theorem 5.5.

Proof. Fix μ to be a near-unanimity operation of arity r under which Γ is closed, and let $\phi = Q_1 v_1 \ldots Q_n v_n \mathcal{C}$ be an instance of the QCSP(Γ) problem. We show that (when $n \geq r$) a new instance ϕ' of the QCSP problem also closed under μ can be created, having the property that it is satisfiable if and only if ϕ is.

The new formula ϕ' is obtained from ϕ by elimination of the innermost quantifier and associated quantified variable, $Q_n v_n$. We split into two cases depending on the type of Q_n.

Case $Q_n = \exists$: Obtain $\mathcal{C}_0$ from $\mathcal{C}$ by establishing r-consistency; this can be done in polynomial time. Since establishing r-consistency can be done using a sequence of projections and intersections, two operations which preserve closure operations, $\mathcal{C}_0$ is closed under μ. Then, create $\mathcal{C}'$ by including in it every constraint from $\mathcal{C}_0$, but "projecting out" the variable v_n from any constraints where it is present. (That is, create $\mathcal{C}'$ as follows: for every constraint $C_0 \in \mathcal{C}_0$, if C_0 does not contain v_n, then include it in $\mathcal{C}'$; otherwise, project C_0 onto its own variable list with v_n excluded, and include the resulting projection in $\mathcal{C}'$.) Define ϕ' to be $Q_1 v_1 \ldots Q_{n-1} v_{n-1} \mathcal{C}'$. Clearly, if ϕ is satisfiable, then ϕ' is satisfiable by the same strategy. If ϕ' is satisfiable, then ϕ is satisfiable by using the same strategy for ϕ' to set $v_1, \ldots, v_{n-1}$; then, Theorem A.4 guarantees that v_n can be set.

Case $Q_n = \forall$: Create $\mathcal{C}'$ by including in it every constraint from $\mathcal{C}$, replacing those constraints $R(\overline{v}) \in \mathcal{C}$ containing the $\forall$-variable v_n with the constraint $S(\overline{w})$ given by rule 2 in Definition 4.10 (with $y = v_n$). It can be verified that $\mathcal{C}'$ is closed under μ. It is straightforward to verify that a winning strategy for ϕ is a wining strategy for $\phi' = Q_1 v_1 \ldots Q_{n-1} v_{n-1} \mathcal{C}'$, and vice-versa. $\square$

[4] We abuse notation here, letting $\overline{v}$ denote not only a tuple as usual, but also the set of variables it contains.