Quantified Constraint Satisfaction and Small Commutative Conservative Operations

Hubie Chen*

July 2003

Abstract

The constraint satisfaction problem (CSP) is a broad framework capturing many combinatorial search problems. A natural and strict generalization of the CSP is the quantified constraint satisfaction problem (QCSP). The CSP involves deciding the truth of constraint networks where all variables are existentially quantified; the QCSP is defined similarly, but variables may be both existentially and universally quantified.

While the CSP and QCSP are in their general formulation intractable, they can be parameterized by restricting the constraint language, that is, the types of constraints that are permitted in problem instances. Much attention has been directed towards classifying the complexity of all constraint languages in the case of the CSP. In this paper, we continue the recently initiated study of QCSP complexity by identifying a new family of tractable constraint languages, namely, constraint languages over domains of small size that are closed under a commutative conservative operation. This gives the first QCSP tractability result based on binary operations which may be non-associative. We also give a complete classification of maximal constraint languages over domains of size three.

1 Introduction

The *constraint satisfaction problem* (CSP) is a general framework in which many combinatorial search 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 their general formulation, the CSP and QCSP are intractable, being NP-complete and PSPACE-complete, respectively; however, it is possible to parameterize these problems by restricting the *constraint language*, or the types of constraints that are permitted in problem instances. This is the form of restriction which was studied by Schaefer in his now classic dichotomy theorem [28], and has seen intense investigation over the past decade in several different contexts. This paper continues the recently initiated study of the complexity of the QCSP under constraint language restrictions [3, 12]. Our main contribution is the identification of a new tractable subclass of the QCSP.

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

1.1 Background

Complexity classification theorems. In 1978, Schaefer proved that every constraint language over a two-element domain gives rise to a CSP subclass which is either in P or is NP-complete [28]. 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 [13], finding the lexicographically minimum satisfying assignment [26], "inverse" satisfiability [22], and finding a satisfying assignment maximizing the number of satisfied constraints [23]. All of these dichotomy theorems are for constraint languages over a two-element domain. Particularly relevant here is the dichotomy theorem for QCSP in domain size two [14, 15], which shows that the only tractable subclasses in this context are QUANTIFIED 2-SAT [1], QUANTIFIED HORN SAT [11], and QUANTIFIED XOR-SAT [14], reflecting exactly the non-trivial tractable constraint languages given by Schaefer's theorem. All other constraint languages give rise to a PSPACE-complete QCSP.

A considerable limitation of the mentioned classification theorems is that they only address constraint languages that are over a two-element domain. Extensions of these theorems that classify all constraint languages over finite domains would be extremely valuable. Such extensions would identify *all* of the ways tractability can arise from constraint language restrictions. Given a restricted subclass of the QCSP that is a candidate for being tractable, one would have the ability to immediately identify whether or not tractability of the subclass can be deduced from its constraint language; if not, it would follow that other features of the subclass need to be utilized in a proof of tractability. Moreover, the intractable subclasses identified by classification theorems can be quite useful in developing further proofs of intractability.

Much attention has recently been directed towards achieving a full classification theorem for the CSP, and has resulted in the identification of many new tractable subclasses of the CSP [24, 18, 20, 7, 16, 10, 5, 17, 6]. A spectacular result produced by this line 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 [28] that had been open for over two decades.

Closure properties and algebra. A powerful *algebraic approach* to studying the relative expressiveness of constraint languages was introduced in [24] and further studied in [19, 20, 21, 9, 7, 16, 10, 4, 5, 6]. In this approach, a dual perspective on the various 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].

Quantified constraint satisfaction. Very recently, the study of QCSP complexity based on constraint languages in domains of size greater than two was initiated independently by Börner, Bulatov, Krokhin and Jeavons [3] and the present author [12]. Both of these works used the algebraic closure properties viewpoint in a central manner. The contributions of [3] included development of the algebraic viewpoint for the QCSP, the identification of some intractable and tractable classes of constraint languages, and a complete complexity classification theorem for a restricted class of constraint languages. The main contribution of [12] was the identification of three broad classes of tractable constraint languages.

1.2 Contributions of this paper

A tractability result. Our main result is the identification of a new family of tractable constraint languages for the QCSP. In particular, we show that for constraint languages on domains of size less than or equal to three, closure under a commutative conservative operation implies QCSP tractability. We prove this for a "multi-sorted" formulation of the QCSP where different variables may have different domains. This gives the first closure property sufficient for QCSP tractability based on binary operations which may be *non-associative*.

Interestingly, we obtain this positive algorithmic result using the same high-level proof technique which was used to show the tractability of constraint languages closed under another form of binary operation, namely, *semilattice operations with bottom* [12].[1] This proof technique involves three steps:

1. Prove a decomposition theorem, showing that all constraints are logically equivalent to the conjunction of constraints in a particular normal form.
2. Give a sound and complete proof system for deriving "implied" constraints from any given quantified formula in the normal form.
3. Construct and prove the correctness of an algorithm for deciding truth based on the proof system.

While the proof outline of our new tractability result is similar to the mentioned previous tractability result, the details of our new result are much more involved. We consider understanding the reach and limitations of this proof technique to be an important issue for future research.

The constraint languages which we show to be tractable for the QCSP were previously shown to be tractable for the CSP [7]. Our new tractability result is further evidence for a trend observed in [12] – namely, that constraint languages that give rise to tractable subclasses of the CSP also tend to give rise to tractable subclasses of the QCSP!

An intractability result. Closure under a semilattice operation was demonstrated in previous work to be a sufficient condition for a constraint language to be CSP-tractable. This result was partially extended to the QCSP: constraint languages closed under a semilattice operation with bottom were shown to be QCSP-tractable in [12], as mentioned. We give a new intractability result, showing that closure under a semilattice operation without bottom is *not* a sufficient condition for a constraint language to be QCSP-tractable. This is particularly significant as it gives the first non-trivial example of a closure condition which guarantees CSP-tractability but does not guarantee QCSP-tractability.[2]

Classification of maximal constraint languages. The final result in this paper is a complete classification of maximal constraint languages over a domain of size three as either tractable or intractable (for the QCSP). A maximal constraint language, intuitively, is a constraint language that can express any constraint with the help of any relation not contained in the language. In other words, they are the largest constraint languages which are not obviously intractable. Identification of tractable maximal constraint languages is extremely important as they are they largest languages that could possibly be tractable – such identification amounts to demonstrating the "broadest" forms of tractability. Consequently, we consider this classification result to be an auspicious indicator of the progress that has been made concerning QCSP complexity thus far. An analogous result for the CSP was presented in previous work [10].

[1] We remark that there is some overlap between the results: any operation isomorphic to the binary operation $\max$ on the set $\{1, \ldots, k\}$ is both a commutative conservative operation and a semilattice operation with bottom.

[2] This claim can be made formal by considering a closure condition to be non-trivial if it is surjective; this is a natural definition in light of the results of [3].

2 Preliminaries

We use the notation $[n]$ as shorthand for the set $\{1, \ldots, n\}$. We use t_i to denote the ith coordinate of a tuple $\overline{t}$. When R is a relation of arity k and $i_1, \ldots, i_m \in [k]$, the notation $\mathrm{pr}_{i_1,\ldots,i_m} R$ denotes the relation $\{(t_{i_1}, \ldots, t_{i_m}) : \overline{t} \in R\}$.

2.1 Quantified Constraint Satisfaction

We now set the basic terminology of quantified constraint satisfaction to be used throughout this paper. The only significant difference between our definitions and those employed in previous work [3, 12] on quantified constraint satisfaction is that we allow quantified formulas to be *multi-sorted*: different variables may have different domains.

Definition 2.1 *A* relation *of* arity $k \geq 1$ *over a collection of sets $\mathcal{A}$ is a subset of $A_1 \times \cdots \times A_k$, where the A_i are taken from $\mathcal{A}$. A* constraint *over $\mathcal{A}$ is an expression of the form $R(\overline{v})$ where R is a relation over $\mathcal{A}$ and $\overline{v}$ is a tuple of variables with the same arity as R. A* constraint language *over $\mathcal{A}$ is a set of relations, all of which are over $\mathcal{A}$, and is said to be finite if it contains finitely many relations.*

Intuitively, a constraint restricts the permissible values of a set of variables; the permissible values are specified by a relation, while the variable subset is specified by a tuple. We consider an arity k constraint $R(\overline{w})$ to be *satisfied* under an interpretation f defined on the variables of $\overline{w}$ if $(f(w_1), \ldots, f(w_k)) \in R$. In this paper, we will only consider relations over collections of finite sets, and throughout we will assume $\mathcal{A}$ to be a collection of finite sets.

Definition 2.2 *A* quantified formula *(over $\mathcal{A}$) is an expression of the form*

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

where:

- *for all $i \in [n]$, Q_i is a quantifier from the set $\{\forall, \exists\}$,*
- *for all $i \in [n]$, v_i is a variable with an associated domain A_i (from $\mathcal{A}$), also denoted by A_{v_i}, and*
- *$\mathcal{C}$ is a finite conjunction of constraints over $\mathcal{A}$ with variables from $\{v_1, \ldots, v_n\}$ such that each constraint $R(v_{i_1}, \ldots, v_{i_k})$ of $\mathcal{C}$, respects the domain A_i of each variable v_i: $R \subseteq A_{i_1} \times \cdots \times A_{i_k}$.*

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 $i < j$, the variable v_j is said to come after the variable v_i in the quantification order of the formula. When Γ is a constraint language, the formula is said to be over Γ if every constraint in $\mathcal{C}$ has relation from Γ.

Truth of a quantified formula is defined just as in first-order logic.

Definition 2.3 *Let Γ be a finite constraint language. The QCSP(Γ) decision problem is to decide, given as input a quantified formula ϕ over Γ, whether or not ϕ is true.*

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

The previous definition allows us to grade the complexity of finite constraint languages Γ according to the complexity of QCSP(Γ). We can extend this grading to infinite constraint languages Γ in the following way.

Definition 2.4 *Let Γ be an infinite constraint language, and let $\mathcal{C}$ be a complexity class. We say that the problem QCSP(Γ) is*

- in $\mathcal{C}$ *if for every finite subset $\Gamma' \subseteq \Gamma$, QCSP(Γ') is in $\mathcal{C}$,*
- $\mathcal{C}$-hard *if there exists a finite subset $\Gamma' \subseteq \Gamma$ such that QCSP(Γ') is $\mathcal{C}$-hard under polynomial-time many-one reductions, and*
- $\mathcal{C}$-complete *if it is both in $\mathcal{C}$ and $\mathcal{C}$-hard.*

The present paper is a contribution to the following long-term research goal.

Open Problem 2.5 *Classify the complexity of QCSP(Γ) for every finite constraint language Γ over a collection of finite sets.*

A classification theorem exists for those constraint languages Γ over a *two*-element domain.

Theorem 2.6 *[14, 15] Suppose that Γ is a finite constraint language over $\{A\}$, where A is a set of size two. Then, QCSP(Γ) is either in P or is PSPACE-complete.*

2.2 Closure Properties

We now define the notion of the *closure properties* of a constraint language, which will give a dual perspective (on constraint languages) relevant to considerations of complexity that has been used to study the CSP [24, 19, 20, 21, 9, 7, 16, 10, 4, 5]. Recall that a constraint language is a set of relations; the closure properties of a constraint language will be a set of functions.

Definition 2.7 *Define f to be* an operation over $\mathcal{A}$ *of arity k if for every A in $\mathcal{A}$, there is an A-interpretation of f, denoted by f^A, which is a mapping from A^k to A.*

That is, an operation over $\mathcal{A}$ is a family of operations of the same arity, each of which is defined on a different set in $\mathcal{A}$. Having precisely defined an operation over $\mathcal{A}$, we can now define what it means for a relation to be closed under an operation.

Definition 2.8 *Let $R \subseteq A_1 \times \cdots \times A_n$ be an n-ary relation over $\mathcal{A}$ and let f be a k-ary operation over $\mathcal{A}$. The relation R is closed under f if for all tuples $\overline{t_1}, \ldots, \overline{t_k} \in R$, the tuple*

$$f(\overline{t_1}, \ldots, \overline{t_k}) \stackrel{\text{def}}{=} (f^{A_1}(t_{11}, \ldots, t_{k1}), \ldots, f^{A_n}(t_{1n}, \ldots, t_{kn}))$$

is also in R.

This notion of closure is straightforwardly extended to constraint languages as follows.

Definition 2.9 *For a constraint language Γ over $\mathcal{A}$, define the* closure properties *or* polymorphisms *of Γ to be the set*

$$\textit{Pol}(\Gamma) = \{f \in \mathcal{O}_{\mathcal{A}} : \text{ every relation in } \Gamma \text{ is closed under } f\}$$

where $\mathcal{O}_{\mathcal{A}}$ denotes the set of all operations over $\mathcal{A}$.

Our interest in the closure properties of a constraint language is based on the following theorem, which was observed for the one-sorted QCSP in [3, 12] and is readily generalized to the multi-sorted QCSP. We use $A \leq_m^p B$ to indicate that there is a polynomial-time many-one reduction from A to B, and $A \equiv_m^p B$ to indicate that $A \leq_m^p B$ and $B \leq_m^p A$.

Theorem 2.10 *Suppose that Γ_1 and Γ_2 are constraint languages over $\mathcal{A}$.*
If Pol$(\Gamma_1) \subseteq$ Pol(Γ_2), then QCSP$(\Gamma_2) \leq_m^p$ QCSP(Γ_1).
If Pol$(\Gamma_1) =$ Pol(Γ_2), then QCSP$(\Gamma_2) \equiv_m^p$ QCSP(Γ_1).

This theorem shows that closure properties are an invariant of complexity: two constraint languages with the same closure properties have exactly the same complexity. The theorem was observed for constraint satisfaction problems in [8] where the closure properties approach to multi-sorted constraints was studied in detail. Essentially the same proof applies to the quantified constraint satisfaction problem. We mention that a stronger version of this theorem for quantified constraint satisfaction problems appears in [3]. We do not need the stronger version for the purposes of this paper, and refer the interested reader thither.

3 Commutative conservative operations

An operation $\star$ over A is *commutative* if it is binary and for all $a, b \in A$, $a \star b = b \star a$. A binary operation $\star$ over A is *conservative* if for all $a, b \in A$, $a \star b \in \{a, b\}$. An operation is *commutative conservative (cc)* if it is both commutative and conservative. An operation $\star$ over a collection of finite sets $\mathcal{A}$ is *commutative conservative* if for every A in $\mathcal{A}$, the operation $\star^A$ (the A-interpretation of $\star$) is commutative conservative.

The following theorem, which we prove in this section, is the main result of this paper.

Theorem 3.1 *Suppose that Γ is a finite constraint language over a collection $\mathcal{A}$ of finite sets, all of which are of size less than or equal to three. If there exists a commutative conservative operation $\star$ over $\mathcal{A}$ under which every relation in Γ is closed, then QCSP(Γ) is decidable in polynomial time.*

As mentioned in the introduction, we prove this theorem in three steps. First, we demonstrate that every constraint closed under a commutative conservative operation can be decomposed into a conjunction of what we call *cc constraints*. Next, we describe a proof system from which further cc constraints can be derived from a quantified formula consisting of cc constraints, and we demonstrate the proof system to be sound and complete. Finally, we give an algorithm which decides the truth of an arbitrary quantified formula (consisting of cc constraints) which makes crucial use of the proof system.

3.1 Decomposition theorem

This subsection establishes a decomposition theorem for constraints with relation closed under a cc operation: every such constraint is logically equivalent to the conjunction of *cc constraints*, constraints of a particular form. When $\oplus$ is a cc operation on a set A, we let $\leq^{A,\oplus}$ denote the relation $\{(a, b) : a \oplus b = b\}$. If $\leq^{A,\oplus}$ is a total ordering, we say that A is *ordered* with respect to $\oplus$; otherwise, we say that A is *unordered* with respect to $\oplus$. When A is ordered with respect to $\oplus$, we let $\max A$ denote the maximal element in the total ordering $\leq^{A,\oplus}$.

We now define cc constraints (relative to a cc operation $\star$ over $\mathcal{A}$) inductively. Expressions of the form T, F, $v \in S$, and $P(v_1, v_2)$, where P is the graph of a bijection closed under $\star$, are all cc constraints. If A is an ordered set with respect to $\star^A$, the containment $a \in A$ holds, and C is a cc constraint, then $v \leq_A a \overset{\wedge}{\Rightarrow} (C)$ is a cc constraint. (Note that when A is a subset of some A' contained in $\mathcal{A}$, we let $\star^A$ denote the restriction of $\star^{A'}$ to A.) We will always omit the parentheses around the embedded constraint C, assuming it to be understood that $\overset{\wedge}{\Rightarrow}$ is right associative.

The semantics of cc constraints are defined as follows. Under the interpretation f,

- the constraint T is true;
- the constraint F is false;

- the constraint $v \in S$ is true if and only if $f(v) \in S$,
- the constraint $P(v_1, v_2)$ is true if and only if $(f(v_1), f(v_2)) \in P$,
- the constraint $v \leq_A a \stackrel{\triangle}{\Rightarrow} C$ is true if and only if
  [$f(v) \in A$ and [$f(v) \leq^{A,\star^A} a$ implies that C is true]].

We say that a cc constraint C is *helpful* if for every C-subexpression of the form $v \leq_A a \stackrel{\triangle}{\Rightarrow} C'$, if A_v is ordered, then $A = A_v$.

Theorem 3.2 *Let R be a relation over a collection $\mathcal{A}$ of finite sets, all of which are of size less than or equal to three. If R is closed under a cc operation, then a constraint with R as relation is logically equivalent to the conjunction of helpful cc constraints.*

3.2 Proof system

We now give a sound and complete proof system for formulas containing only cc constraints. Without loss of generality, we focus on quantified formulas of the form $\forall y_1 \exists x_1 \dots \forall y_n \exists x_n \, \mathcal{C}(y_1, x_1, \dots, y_n, x_n)$. A *strategy* for such a formula is a sequence of mappings $\tau_1 : A_{y_1} \to A_{x_1}, \dots, \tau_n : A_{y_1} \times \cdots A_{y_n} \to A_{x_n}$, and is a *winning strategy* if for all mappings f defined on $\{y_1, \dots, y_n\}$ with $f(y_i) \in A_{y_i}$, the assignment mapping x_i to $\tau_i(f(y_1), \dots, f(y_i))$ and y_i to $f(y_i)$ (for all $i \in [n]$) satisfies every constraint in $\mathcal{C}$. Note that a quantified formula is satisfiable if and only if there exists a winning strategy for the formula. We say that a proof system $\vdash$ is *sound* if for all formulas ϕ and constraints C such that $\phi \vdash C$, every winning strategy for ϕ satisfies C. We say that a proof system $\vdash$ is *complete* if for all unsatisfiable formulas ϕ, the constraint F is derivable from ϕ: $\phi \vdash F$.

In the specification of the proof system, the letter x denotes a $\exists$-variable, the letter y denotes a $\forall$-variable, and the letters v, w denote variables of either type. An *x-unit constraint* (in a formula ϕ) is a cc constraint $C = \alpha_1 \stackrel{\triangle}{\Rightarrow} \cdots \stackrel{\triangle}{\Rightarrow} \alpha_l \stackrel{\triangle}{\Rightarrow} \beta$ such that all variables in the α_i's are $\forall$-variables, the variable x occurs in β, and no variable occurring in C comes after x in the quantification order of ϕ. In other words, all of the α_i's are over $\forall$-variables coming before x, and β is either of the form $x \in S$ or $P(v, x)$ for some variable v coming before x. The notation $C : u(x)$ is used to indicate that C is an x-unit constraint. We say that a constraint is a *unit constraint* if it is an x-unit constraint for some $\exists$-variable x. An x-unit constraint $C = \alpha_1 \stackrel{\triangle}{\Rightarrow} \cdots \stackrel{\triangle}{\Rightarrow} \alpha_l \stackrel{\triangle}{\Rightarrow} \beta$ is an *x-set constraint* if β is of the form $x \in S$, and is an *x-permutation constraint* if β is of the form $P(v, x)$ with $v \neq x$.

We assume it to be understood that when a cc constraint $\alpha_1 \stackrel{\triangle}{\Rightarrow} \cdots \stackrel{\triangle}{\Rightarrow} \alpha_l \stackrel{\triangle}{\Rightarrow} P(v, w)$ is derivable, the essentially equivalent cc constraint $\alpha_1 \stackrel{\triangle}{\Rightarrow} \cdots \stackrel{\triangle}{\Rightarrow} \alpha_l \stackrel{\triangle}{\Rightarrow} P^{-1}(w, v)$ is also derivable, where P^{-1} denotes the relation $\{(b, a) : (a, b) \in P\}$. The notation $P \circ P'$ is used to denote the relation $\{(a, c) : \exists b$ such that $(a, b) \in P$ and $(b, c) \in P'\}$. The notation $C \in \phi$ indicates that the constraint C is an element appearing in the quantified formula ϕ.

The following are the rules of our proof system; all rules except the first have associated names.

1. $\dfrac{C \in \phi}{\phi \vdash C}$

2. Self-permutation

   $$\frac{\phi \vdash \alpha_1 \stackrel{\triangle}{\Rightarrow} \cdots \stackrel{\triangle}{\Rightarrow} \alpha_l \stackrel{\triangle}{\Rightarrow} P(x, x) : u(x)}{\phi \vdash \alpha_1 \stackrel{\triangle}{\Rightarrow} \cdots \stackrel{\triangle}{\Rightarrow} \alpha_l \stackrel{\triangle}{\Rightarrow} x \in \{a : P(a, a)\}}$$

3. Intersection

$$\frac{\phi \vdash \alpha_1 \stackrel{\triangle}{\Rightarrow} \cdots \stackrel{\triangle}{\Rightarrow} \alpha_l \stackrel{\triangle}{\Rightarrow} x \in S : u(x) \quad \phi \vdash \alpha'_1 \stackrel{\triangle}{\Rightarrow} \cdots \stackrel{\triangle}{\Rightarrow} \alpha'_m \stackrel{\triangle}{\Rightarrow} x \in S' : u(x)}{\phi \vdash \alpha_1 \stackrel{\triangle}{\Rightarrow} \cdots \stackrel{\triangle}{\Rightarrow} \alpha_l \stackrel{\triangle}{\Rightarrow} \alpha'_1 \stackrel{\triangle}{\Rightarrow} \cdots \stackrel{\triangle}{\Rightarrow} \alpha'_m \stackrel{\triangle}{\Rightarrow} x \in S \cap S'}$$

4. Empty-to-false

$$\frac{\phi \vdash \alpha_1 \stackrel{\triangle}{\Rightarrow} \cdots \stackrel{\triangle}{\Rightarrow} \alpha_l \stackrel{\triangle}{\Rightarrow} x \in \emptyset : u(x)}{\phi \vdash \alpha_1 \stackrel{\triangle}{\Rightarrow} \cdots \stackrel{\triangle}{\Rightarrow} \alpha_l \stackrel{\triangle}{\Rightarrow} F}$$

5. Existential permutation

$$\frac{\phi \vdash \alpha_1 \stackrel{\triangle}{\Rightarrow} \cdots \stackrel{\triangle}{\Rightarrow} \alpha_l \stackrel{\triangle}{\Rightarrow} P(v,x) : u(x) \quad v \neq x \quad \phi \vdash \alpha'_1 \stackrel{\triangle}{\Rightarrow} \cdots \stackrel{\triangle}{\Rightarrow} \alpha'_m \stackrel{\triangle}{\Rightarrow} x \in S : u(x)}{\phi \vdash \alpha_1 \stackrel{\triangle}{\Rightarrow} \cdots \stackrel{\triangle}{\Rightarrow} \alpha_l \stackrel{\triangle}{\Rightarrow} \alpha'_1 \stackrel{\triangle}{\Rightarrow} \cdots \stackrel{\triangle}{\Rightarrow} \alpha'_m \stackrel{\triangle}{\Rightarrow} v \in \{a : \exists s \in S \text{ such that } P(a,s)\}}$$

$$\frac{\begin{array}{c}\phi \vdash \alpha_1 \stackrel{\triangle}{\Rightarrow} \cdots \stackrel{\triangle}{\Rightarrow} \alpha_l \stackrel{\triangle}{\Rightarrow} \beta \quad \alpha_j = (x \leq_A a) \quad \phi \vdash \alpha'_1 \stackrel{\triangle}{\Rightarrow} \cdots \stackrel{\triangle}{\Rightarrow} \alpha'_m \stackrel{\triangle}{\Rightarrow} P(v,x) : u(x) \\ P(b,a) \quad B = \{b' : \exists a' \in A \text{ such that } P(b',a')\}\end{array}}{\phi \vdash \alpha_1 \stackrel{\triangle}{\Rightarrow} \cdots \stackrel{\triangle}{\Rightarrow} \alpha_{j-1} \stackrel{\triangle}{\Rightarrow} \alpha'_1 \stackrel{\triangle}{\Rightarrow} \cdots \stackrel{\triangle}{\Rightarrow} \alpha'_m \stackrel{\triangle}{\Rightarrow} v \leq_B b \stackrel{\triangle}{\Rightarrow} \alpha_{j+1} \stackrel{\triangle}{\Rightarrow} \cdots \stackrel{\triangle}{\Rightarrow} \alpha_l \stackrel{\triangle}{\Rightarrow} \beta}$$

6. Permutation join

$$\frac{\phi \vdash \alpha_1 \stackrel{\triangle}{\Rightarrow} \cdots \stackrel{\triangle}{\Rightarrow} \alpha_l \stackrel{\triangle}{\Rightarrow} P(v,x) : u(x) \quad \phi \vdash \alpha'_1 \stackrel{\triangle}{\Rightarrow} \cdots \stackrel{\triangle}{\Rightarrow} \alpha'_m \stackrel{\triangle}{\Rightarrow} P'(x,w) : u(x)}{\phi \vdash \alpha_1 \stackrel{\triangle}{\Rightarrow} \cdots \stackrel{\triangle}{\Rightarrow} \alpha_l \stackrel{\triangle}{\Rightarrow} \alpha'_1 \stackrel{\triangle}{\Rightarrow} \cdots \stackrel{\triangle}{\Rightarrow} \alpha'_m \stackrel{\triangle}{\Rightarrow} (P \circ P')(v,w)}$$

7. Universal elimination

Suppose that $\phi \vdash C$, $C = \alpha_1 \stackrel{\triangle}{\Rightarrow} \cdots \stackrel{\triangle}{\Rightarrow} \alpha_l \stackrel{\triangle}{\Rightarrow} \beta$ and that y is a variable in C which is after all other variables in C relative to the quantification order of ϕ. If there exists a setting $y = d$ such that for every $\alpha_j = (v_j \leq_{A_j} a_j)$ with variable $v_j = y$, it is the case that $d \in A_j$ and $d \leq^{A_j,\star^{A_j}} a_j$, then

$$\phi \vdash \alpha_{i_1} \stackrel{\triangle}{\Rightarrow} \cdots \stackrel{\triangle}{\Rightarrow} \alpha_{i_m} \stackrel{\triangle}{\Rightarrow} \beta'$$

where $\alpha_{i_1}, \ldots, \alpha_{i_m}$ is the subsequence of $\alpha_1, \ldots, \alpha_l$ obtained by removing all α_j such that $v_j = y$, and β' is:

- T if $\beta = T$ or [$\beta = (y \in S)$ and $d \in S$]
- F if $\beta = F$ or [$\beta = (y \in S)$ and $d \notin S$]
- $(v \in \{d'\})$ if $\beta = P(y,v)$ and $P(d,d')$ is true.

8. Prefixing

$$\frac{\phi \vdash \alpha_1 \stackrel{\triangle}{\Rightarrow} \cdots \stackrel{\triangle}{\Rightarrow} \alpha_l \stackrel{\triangle}{\Rightarrow} \beta \quad \alpha_i = (v \leq_A a)}{\phi \vdash \alpha_1 \stackrel{\triangle}{\Rightarrow} \cdots \stackrel{\triangle}{\Rightarrow} \alpha_{i-1} \stackrel{\triangle}{\Rightarrow} v \in A}$$

When the prefixing rule is applicable, we call the conclusion a *prefix* of the premise.

9. Resolution

$$\frac{\begin{array}{c}\phi \vdash \alpha_1 \stackrel{\triangle}{\Rightarrow} \cdots \stackrel{\triangle}{\Rightarrow} \alpha_l \stackrel{\triangle}{\Rightarrow} \beta \quad \alpha_j = (x \leq_A a) \quad \alpha'_1 \stackrel{\triangle}{\Rightarrow} \cdots \stackrel{\triangle}{\Rightarrow} \alpha'_m \stackrel{\triangle}{\Rightarrow} x \in S : u(x) \\ S \text{ ordered} \quad S \subseteq A \quad \max S \leq^{A,\star^A} a\end{array}}{\phi \vdash \alpha_1 \stackrel{\triangle}{\Rightarrow} \cdots \stackrel{\triangle}{\Rightarrow} \alpha_{j-1} \stackrel{\triangle}{\Rightarrow} \alpha'_1 \stackrel{\triangle}{\Rightarrow} \cdots \stackrel{\triangle}{\Rightarrow} \alpha'_m \stackrel{\triangle}{\Rightarrow} \alpha_{j+1} \stackrel{\triangle}{\Rightarrow} \cdots \stackrel{\triangle}{\Rightarrow} \alpha_l \stackrel{\triangle}{\Rightarrow} \beta}$$

Theorem 3.3 *The given proof system is sound and complete for the class of formulas containing only cc constraints.*

Proof. Soundness is straightforward to establish, so we prove completeness. Suppose that the false constraint F is not derivable from $\phi = \forall y_1 \exists x_1 \dots \forall y_n \exists x_n\, \mathcal{C}(y_1, x_1, \dots, y_n, x_n)$. We prove that ϕ is satisfiable by describing a winning strategy, inductively. Define $S_k(a_1, \dots, a_k)$ to be the set

$$\{a : \tau(a_1, \dots, a_k)[x_k = a] \text{ satisfies all derivable } x_k\text{-unit constraints}\}$$

where $\tau(a_1, \dots, a_k)$ is defined to be the mapping taking y_i to a_i for all $i \in [k]$, and x_i to $\tau_i(a_1, \dots, a_i)$ for all $i \in [k-1]$. The strategy is defined as follows: if $S_k(a_1, \dots, a_k)$ is ordered with respect to $\star$ on A_{x_k}, set $\tau_k(a_1, \dots, a_k)$ to be equal to $\max S_k(a_1, \dots, a_k)$; otherwise, set $\tau_k(a_1, \dots, a_k)$ arbitrarily.

We first show that this strategy is well-defined. Precisely, we show that if there exists a setting $a_1, \dots, a_k$ to $y_1, \dots, y_k$ such that $S_k(a_1, \dots, a_k)$ is empty, then F is derivable. We prove this by contradiction; suppose that there is such a setting, and that k is the minimal value with such a setting. Then, for each $a \in A_{x_k}$, there is a x_k-unit constraint C_a that is false under $\tau(a_1, \dots, a_k)[x_k = a]$. We assume (for all $a \in A_{x_k}$) that every prefix of C_a is true under $\tau(a_1, \dots, a_k)$ (if this assumption fails, then prefixing and universal elimination can be applied to derive F). By the self-permutation rule, we can assume that each C_a is an x_k-permutation constraint or an x_k-set constraint. All of the C_a that are x_k-set constraints can be combined, by the intersection rule, to derive a "most stringent" set constraint C such that C is false under $\tau(a_1, \dots, a_k)[x_k = a]$ for all values a such that C_a is a set constraint. The constraint C must be of the form $\alpha_1 \overset{\wedge}{\Rightarrow} \cdots \overset{\wedge}{\Rightarrow} \alpha_l \overset{\wedge}{\Rightarrow} x \in S$. We break into two cases depending on whether or not S is empty.

Case 1: $S = \emptyset$. By a single application of the empty-to-false rule followed by (possibly repeated) application of the universal elimination rule (using the values a_i for the y_i), the constraint F can be derived from C.

Case 2: $S \neq \emptyset$. Fix $b \in S$. We have that C_b is a permutation constraint; let d be the unique value of A_{x_k} such that $\tau(a_1, \dots, a_k)[x_k = d]$ satisfies C_b.

If $d \in S$, then applying the permutation join rule to C_b and C_d allows us to derive a permutation constraint that is false under $\tau(a_1, \dots, a_k)$, contradicting the definition of τ.

If $d \notin S$, then apply the first form of the existential permutation rule to C and C_b to obtain a constraint D. The constraint D may be in one of two forms. It may be a constraint with only $\forall$-variables that is false under the assignment taking y_i to a_i for $i \in [k]$, in which case universal elimination can be repeatedly applied to derive F. Otherwise, D is a constraint of the form $\alpha_1 \overset{\wedge}{\Rightarrow} \cdots \overset{\wedge}{\Rightarrow} \alpha_l \overset{\wedge}{\Rightarrow} x' \in S'$ such that only $\forall$-variables occur in the α_i's, x' comes before x in the quantification order of ϕ, and D is false under $\tau(a_1, \dots, a_k)$. Using universal elimination to eliminate the $\forall$-variables in D which come after x', we obtain a unit constraint false under $\tau(a_1, \dots, a_k)$, contradicting the definition of τ.

This concludes the demonstration that the described strategy is well-defined. We now show that the described strategy is a winning strategy. In particular, we prove that every constraint $C = \alpha_1 \overset{\wedge}{\Rightarrow} \cdots \overset{\wedge}{\Rightarrow} \alpha_l \overset{\wedge}{\Rightarrow} \beta$ derivable from ϕ is satisfied by the strategy, by induction on t, the number of α_i's which contain $\exists$-variables.

Base case: $t = 0$. If C consists only of $\forall$-variables and is false under some assignment $a_1, \dots, a_n$ to $y_1, \dots, y_n$, then F can be derived from C by a single use of the prefixing rule, followed by (possibly repeated) application of the universal elimination rule. Otherwise, β contains at least one $\exists$-variable and we can assume that every prefix of C is true under any assignment $a_1, \dots, a_n$ to $y_1, \dots, y_n$. If C is false under $\tau(a_1, \dots, a_k)$ for some values $a_1, \dots, a_n$, then universal elimination (with respect to the a_i) can be used to eliminate all $\forall$-variables in C after the $\exists$-variables, yielding a unit constraint C' which is still false under the assignment taking $y_1, \dots, y_n$ to $a_1, \dots, a_n$. However, C' is a unit constraint and is satisfied by the strategy τ by definition, contradicting the assumption that C was false.

Induction case: $t \geq 1$. Suppose for a contradiction that there is an assignment $a_1, \dots, a_n$ to $y_1, \dots, y_n$ such that C is false under $\tau(a_1, \dots, a_n)$. We can assume without loss of generality that all prefixes of C are true under $\tau(a_1, \dots, a_n)$, and that x_k is an $\exists$-variable appearing in some α_j of C.

Suppose that there is a derivable x_k-permutation constraint E, such that there is a unique value $a \in A_{x_k}$ having the property that E is true under $\tau(a_1, \ldots, a_k)[x_k = a]$. ¿From E and C, the existential permutation rule can be used to derive a constraint C' false under $\tau(a_1, \ldots, a_n)$ with either a lower t value than that of C, contradicting the induction hypothesis; or with $\exists$-variables whose indices come before those of C in a lexicographic ordering, contradicting a suitably constructed induction hypothesis.

Suppose that there is no such derivable x_k-permutation constraint; let $E = \alpha'_1 \overset{\wedge}{\Rightarrow} \cdots \overset{\wedge}{\Rightarrow} \alpha'_m \overset{\wedge}{\Rightarrow} x_k \in S$ be the most stringent x_k-set constraint under the assignment $a_1, \ldots, a_k$ to $y_1, \ldots, y_k$, and suppose that α_j is equal to $x_k \leq_A a$. Since all prefixes of C are true under $\tau(a_1, \ldots, a_n)$ but C itself is false under $\tau(a_1, \ldots, a_n)$, we have $S \subseteq A$ and $(\max S) \leq^{A, \star^A} a$. Hence the resolution rule can be used to derive, from E and C, an unsatisfiable constraint with a lower value of t, contradicting the induction hypothesis. $\square$

3.3 Algorithm

We now turn to the proof of Theorem 3.1. We are given a quantified formula ϕ with constraints closed under a commutative conservative operation. By appeal to Theorem 3.2, we assume that the constraints of ϕ, which we denote by $\mathcal{C}$, are all helpful cc constraints. We assume without loss of generality that for every constraint $R(v_1, \ldots, v_m)$ in the original formula, $\mathrm{pr}_i R = A_{v_i}$, so that for every subexpression $y \leq_A a \overset{\wedge}{\Rightarrow} C$ of a constraint in $\mathcal{C}$, A is equal to A_y.

We rewrite the constraints $\mathcal{C}$ as a new conjunction of constraints $\mathcal{C}'$ as follows. For every constraint C in $\mathcal{C}$, there is a constraint C' in $\mathcal{C}'$. If $C = \alpha_1 \overset{\wedge}{\Rightarrow} \cdots \alpha_l \overset{\wedge}{\Rightarrow} \beta$ is a constraint in $\mathcal{C}$ containing an α_i over a $\forall$-variable y such that A_y is unordered, let j be the index of the first such α_i, and set $C' = \alpha_1 \overset{\wedge}{\Rightarrow} \cdots \overset{\wedge}{\Rightarrow} \alpha_{j-1} \overset{\wedge}{\Rightarrow} F$. Likewise, if there is no α_i of the described form but β is of the form $y \in S$ for some proper subset S of A_y, and A_y is unordered, then set $C' = \alpha_1 \overset{\wedge}{\Rightarrow} \cdots \overset{\wedge}{\Rightarrow} \alpha_l \overset{\wedge}{\Rightarrow} F$. Otherwise, simply set $C' = C$. We claim that ϕ is satisfiable if and only if $\mathcal{C}'$ is satisfiable (where $\mathcal{C}'$ is understood to be shorthand for the quantified formula with the same quantifier prefix as ϕ and constraints $\mathcal{C}'$). Any winning strategy for ϕ can be verified to be a winning strategy for $\mathcal{C}'$, so if $\mathcal{C}'$ is satisfiable, then $\mathcal{C}$ is satisfiable. If $\mathcal{C}'$ is unsatisfiable, then F is derivable from $\mathcal{C}'$; it follows that either F or an expression of the form $y \in S$, with S a proper subset of A_y, is derivable from ϕ (via a derivation of similar structure), implying that ϕ is unsatisfiable.

Next, let $\mathcal{C}''$ be the closure of $\mathcal{C}'$ under the prefixing rule. We claim that F is derivable from $\mathcal{C}'$ using all derivation rules if and only if F is derivable from $\mathcal{C}''$ using all derivation rules except for prefixing. The nontrivial direction of this claim is the forward direction. It can be shown by examining each rule that for any constraint $C = \alpha_1 \overset{\wedge}{\Rightarrow} \cdots \overset{\wedge}{\Rightarrow} \alpha_l \overset{\wedge}{\Rightarrow} \beta$, derivable from $\mathcal{C}'$, either C is derivable from $\mathcal{C}''$ without use of prefixing, or a "stronger" version of C is derivable from $\mathcal{C}''$ without use of prefixing, where $\alpha'_1 \overset{\wedge}{\Rightarrow} \cdots \overset{\wedge}{\Rightarrow} \alpha'_m \overset{\wedge}{\Rightarrow} \beta$ is considered to be stronger than C if $\alpha'_1, \ldots, \alpha'_m$. is a subsequence of $\alpha_1, \ldots, \alpha_l$.

We have established that the original formula ϕ is satisfiable if and only if F is derivable from $\mathcal{C}''$ without use of the prefixing rule. Let $\mathcal{D}$ be those constraints in $\mathcal{C}''$ ending with $y \in S$ or F and $\mathcal{U}$ be all other constraints in $\mathcal{C}''$. It can be shown that if F is derivable from $\mathcal{C}''$, then F is derivable from $\mathcal{U} \cup \{D\}$ for some $D \in \mathcal{D}$. In other words, if F can be derived at all from $\mathcal{C}''$, then only one constraint in $\mathcal{D}$ is required in a derivation of F. Roughly, this is because the non-trivial constraints derivable from a constraint $D \in \mathcal{D}$ are those obtained through use of the resolution rule, which can only be used to combine D with unit constraints, that is, constraints outside of $\mathcal{D}$.

It thus suffices to show how to decide a formula with constraints of the form $\mathcal{U} \cup \{D\}$. Such a formula can be simplified by setting all $\forall$-variables y over an ordered domain (that is, such that A_y is ordered) – other than that possibly appearing in the β portion of D – to the minimal element in the total ordering $\leq^{A_y, \star^{A_y}}$. It can be shown that if there is a winning strategy for the $\forall$-player, there is one where he sets those variables as such. Other than possibly the $\forall$-variable appearing in the β portion of D, the $\forall$-variables in the simplified formula all occur inside permutations $P(v_1, v_2)$. Hence the only unit constraints are of length two, that is, of the form $\alpha \overset{\wedge}{\Rightarrow} \beta$. Moreover, it can be verified that no application of proof system rules to the simplified formula can increase the length of a constraint by more than one; roughly, this is because any rule which combines two constraints together requires one of the constraints to be a unit constraint. Since the original constraint language Γ was fixed, all derivable constraints are of length bounded above by a constant, and so there are polynomially many derivable constraints. All derivable constraints can thus be computed in polynomial time, and hence whether or not F is derivable is decidable in polynomial time.

4 Semilattice operations without bottoms

In this section, we show that closure under a semilattice operation without bottom is *not* sufficient to guarantee tractability of a constraint language. Recall that a *semilattice operation* is a binary operation that is associative, commutative, and idempotent. Any semilattice operation $\oplus$ on a set A induces a partial ordering $\leq_{\oplus}$ defined by $a \leq_{\oplus} b \Leftrightarrow a \oplus b = b$. We say that a semilattice operation $\oplus$ on a set A is a *semilattice operation with bottom* if there exists an element $\mathbf{b} \in A$ such that for all $a \in A$, $\mathbf{b} \leq_{\oplus} a$, and is a *semilattice operation without bottom* otherwise.

Semilattice operations with bottom were shown to guarantee tractability of the QCSP in [12]. The following theorem complements this tractability result, showing that the remaining semilattice operations – those without bottom – do not guarantee tractability of the QCSP.

Theorem 4.1 *Let $\oplus$ be a semilattice operation without bottom on a finite set A, and let Γ be the constraint language containing all relations over $\{A\}$ closed under $\oplus$. The problem QCSP(Γ) is coNP-hard, even when restricted to quantifier prefix $\forall\exists$.*

Note that this theorem can be extended in a straightforward way to multi-sorted constraint languages closed under a semilattice operation.

Proof. Assuming that $\oplus$ is a semilattice operation without bottom on the finite set A, it can be shown that there exist distinct elements $b_0, b_1 \in A$ such that b_0 and b_1 are both minimal with respect to $\leq_{\oplus}$, that is, for all $a \in A$ and $i \in \{0, 1\}$, $a \leq_{\oplus} b_i$ implies $a = b_i$.

We show coNP-hardness by reducing from the propositional tautology problem. Let $C(y_1, \ldots, y_n)$ be an instance of this problem, where C is a circuit with input gates having labels $y_1, \ldots, y_n$. We assume without loss of generality that all non-input gates of C are either AND or NOT gates, and assign all non-input gates labels $x_1, \ldots, x_m$. The quantifier prefix of the resulting quantified formula is $\forall y_1 \ldots \forall y_n \exists x_1 \ldots \exists x_m$. The conjunction of constraints in the resulting formula is constructed as follows.

For each AND gate x_i with inputs $v, v' \in \{y_1, \ldots, y_n\} \cup \{x_1, \ldots, x_m\}$, include the four constraints:

- $v \leq_{\oplus} b_1 \wedge v' \leq_{\oplus} b_1 \Rightarrow x_i \leq_{\oplus} b_1$
- $v \leq_{\oplus} b_0 \wedge v' \leq_{\oplus} b_1 \Rightarrow x_i \leq_{\oplus} b_0$
- $v \leq_{\oplus} b_1 \wedge v' \leq_{\oplus} b_0 \Rightarrow x_i \leq_{\oplus} b_0$
- $v \leq_{\oplus} b_0 \wedge v' \leq_{\oplus} b_0 \Rightarrow x_i \leq_{\oplus} b_0$

For each NOT gate x_i with input $v \in \{y_1, \ldots, y_n\} \cup \{x_1, \ldots, x_m\}$, include the two constraints:

- $v \leq_{\oplus} b_0 \Rightarrow x_i \leq_{\oplus} b_1$
- $v \leq_{\oplus} b_1 \Rightarrow x_i \leq_{\oplus} b_0$

For the output gate x_o, include the constraint

- $x_o \leq_\oplus b_0 \Rightarrow$ FALSE

Note that it is straightforward to verify that all of the given constraints are closed under $\oplus$. We verify the reduction to be correct as follows.

Suppose that the original circuit was a tautology. Let $f : \{y_1, \ldots, y_n\} \to A$ be any assignment to the $\forall$-variables of the quantified formula. Define $f' : \{y_1, \ldots, y_n\} \to \{b_0, b_1\}$ by $f'(y_i) = f(y_i)$ if $f(y_i) \in \{b_0, b_1\}$, and b_0 otherwise. (The value b_0 is an arbitrary choice here; b_1 works just as well.) The AND and NOT gate constraints force each x_i to have either the value b_0 or b_1 under f'; it can be verified that the assignment taking x_i to its forced value and y_i to $f(y_i)$ satisfies all of the constraints.

Suppose that the original circuit was not a tautology. Let $g : \{y_1, \ldots, y_n\} \to \{0, 1\}$ be an assignment making the circuit C false. Let $g' : \{y_1, \ldots, y_n\} \to \{b_0, b_1\}$ be the assignment to the $\forall$-variables of the quantified formula defined by $g'(y_i) = b_{g(y_i)}$. Under g', the only assignment to the x_i variables satisfying the AND and NOT gate constraints is the mapping taking x_i to b_0 if the gate x_i has value 0 under g, and b_1 if the gate x_i has value 1 under g. Because of the output gate constraint, no assignment to the x_i variables satisfies all of the constraints under g', and so the quantified formula is unsatisfiable. $\square$

5 Maximal constraint languages

Throughout this section, we only consider constraint languages over a single finite set. A closure property is a *projection* if it is an arity k operation of the form $f(a_1, \ldots, a_k) = a_i$, for some $i \in [k]$. We say that a closure property is *nontrivial* if it is not a projection.

Definition 5.1 *Let A be a finite set. A constraint language Γ over A is* maximal *if it has a nontrivial closure property f and for all relations R (over A) not in Γ, R is not closed under f.*

A theorem of Rosenberg [27] demonstrates that all maximal constraint languages are closed under an operation of a particular form. In order to state this theorem, we require some new terminology. An operation $f : A^k \to A$ is a *majority* operation if $k = 3$ and for all $a, a' \in A$ the equalities $f(a, a, a') = f(a, a', a) = f(a', a, a) = a$ hold; an *affine* operation if $k = 3$ and for all $a_1, a_2, a_3 \in A$ it is the case that $f(a_1, a_2, a_3) = a_1 * a_2^{-1} * a_3$ where $*$ is a binary operation and $^{-1}$ is a unary operation such that $(A, *, ^{-1})$ is an Abelian group; and a *semiprojection* if $k \geq 3$, f is not a projection, and there exists $i \in [k]$ such that for all $a_1, \ldots, a_k \in A$, $|\{a_1, \ldots, a_k\}| < k$ implies $f(a_1, \ldots, a_k) = a_i$.

Theorem 5.2 *[27] If Γ is a maximal constraint language, then Γ is exactly the set of relations closed under an operation of one of the following types:*

1. *a unary operation which is either a bijection or acts identically on its range,*
2. *a semiprojection,*
3. *a binary idempotent operation which is not a projection,*
4. *a majority operation,*
5. *an affine operation.*

Before giving our classification theorem for maximal constraint languages, we establish the following lemma, which conerns the first two types of Theorem 5.2.

Lemma 5.3 *Let Γ be a maximal constraint language that is exactly the set of relations closed under f, where f is either*

- *a unary operation which is either a bijection or acts identically on its range, or*
- *a semiprojection.*

Then QCSP(Γ) *is PSPACE-complete.*

Proof. By the QCSP classification theorem for domain size two, the the not-all-equal relation $\{0,1\}^3 \setminus \{(0,0,0),(1,1,1)\}$ is intractable [14, 15]; in addition, the not-equal relation $\{(a,b) \in D^2 : a \neq b\}$ for domains D of size 3 was shown to be intractable in [3]. If f is not a constant, then by reasoning similar to that in [10, Lemma 1] and [10, Corollary 1], along with the mentioned intractability results, we obtain that QCSP(Γ) is PSPACE-complete. If f is a constant, then it can be verified that any constraint can be simulated by a quantified formula over Γ with free variables, from which the PSPACE-completeness of QCSP(Γ) follows immediately. $\square$

Using the given theorem and lemma, we can now establish our classification theorem.

Theorem 5.4 *Let* Γ *be a maximal constraint language over a finite set of size three. One of the following three possibilities holds: QCSP*(Γ) *is in P, QCSP*(Γ) *is PSPACE-complete, or* Γ *is exactly the set of relations closed under a semilattice operation with bottom, in which case QCSP*(Γ) *is coNP-hard.*

For maximal constraint languages Γ of the third type (that is, sets of relations closed under a semilattice operation with bottom), describing the exact complexity of QCSP(Γ) is an intriguing open question.

Proof. We consider the five cases of Theorem 5.2. If one of the first two cases holds, then QCSP(Γ) is PSPACE-complete by Lemma 5.3. If Γ is closed under a majority operation or an affine operation, then QCSP(Γ) is in P by the tractability results in [12]. It remains to consider maximal constraint languages clsoed under binary idempotent operations. By the analysis in [10, Section 3.2], one of three possibilities holds: Γ is closed under an affine operation, Γ is closed under a commutative conservative operation, or Γ is exactly the set of relations closed under a semilattice operation. For the first two of these three possibilities, the fact that QCSP(Γ) is in P follows from a tractability result in [12] and Theorem 3.1, respectively. If the third possibility holds and the semilattice operation is one with bottom, then QCSP(Γ) is in P by a tractability result in [12]; otherwise, QCSP(Γ) is coNP-hard by Theorem 4.1. $\square$

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] F. Börner, A. Bulatov, A. Krokhin, and P. Jeavons. Quantified constraints: Algorithms and complexity. In *Computer Science Logic 2003*, 2003.

[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. Tractable conservative constraint satisfaction problems. In *Proceedings of 18th IEEE Symposium on Logic in Computer Science (LICS '03)*, pages 321–330, 2003.

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

[8] Andrei A. Bulatov and Peter Jeavons. An algebraic approach to multi-sorted constraints. Technical Report PRG-RR-01-18, Oxford University, 2001.

[9] 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.

[10] 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.

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

[12] Hubie Chen. Quantified constraint satisfaction problems: Closure properties, complexity, and proof systems. Cornell technical report, 2003.

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

[14] 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.

[15] 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.

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

[17] 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.

[18] 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.

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

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

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

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

[23] 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.

[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] I.G. Rosenberg. Minimal clones I: the five types. In *Lectures in Universal Algebra (Proc. Conf. Szeged 1983)*, volume 43 of *Colloq. Math. Soc. Janos Bolyai*, pages 405–427. North-Holland, 1986.

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