Electronic Colloquium on Computational Complexity, Report No. 36 (2005)

Quantified Constraint Satisfaction, Maximal Constraint Languages, and Symmetric Polymorphisms

Hubie Chen

Departament de Tecnologia
Universitat Pompeu Fabra
Barcelona, Spain
hubie.chen@upf.edu

Abstract. The constraint satisfaction problem (CSP) is a convenient framework for modelling search problems; the CSP involves deciding, given a set of constraints on variables, whether or not there is an assignment to the variables satisfying all of the constraints. This paper is concerned with the quantified constraint satisfaction problem (QCSP), a more general framework in which variables can be quantified both universally and existentially. We study the complexity of restricted cases of the QCSP where the types of constraints that may appear are restricted by a constraint language. We give a complete complexity classification of maximal constraint languages, the largest possible languages that can be tractable. We also give a complete complexity classification of constraint languages arising from symmetric polymorphisms.

1 Introduction

The *constraint satisfaction problem (CSP)* is widely acknowledged as a convenient framework for modelling search problems. An instance of the CSP consists of a set of variables, a domain, and a set of constraints; each constraint consists of a tuple of variables paired with a relation (over the domain) which contains permitted values for the variable tuple. The question is to decide whether or not there is an assignment mapping each variable to a domain element that satisfies all of the constraints. Alternatively, the CSP may be viewed as the problem of deciding, given an ordered pair of relational structures, whether or not there exists a homomorphism from the first structure to the second. Canonical examples of CSPs include boolean satisfiability and graph coloring problems.

All of the variables in a CSP can be viewed as being implicitly existentially quantified. A natural and useful generalization of the CSP is the *quantified constraint satisfaction problem (QCSP)*, where variables may be both universally and existentially quantified. An instance of the QCSP can be viewed as a game between two players which take turns setting variables occurring in a set of constraints; the question is to decide whether or not a specified player can always succeed in satisfying all of the constraints, despite the moves of the other

ISSN 1433-8092

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 is a prototypical PSPACE reasoning problem which captures the complexity of many problems involving *interaction* among multiple agents. Such problems arise naturally in a wide variety of domains, for example, combinatorics, logic, game theory, and artificial intelligence.

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 [30], 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, 13, 14]. Our contributions are the complete classification of *maximal constraint languages*, as well as the complete classification of idempotent *symmetric polymorphisms*.

1.1 Background

Complexity classification theorems. In 1978, Schaefer proved that every constraint language over a two-element domain gives rise to a case of the CSP that is either in P or is NP-complete [30]. The non-trivial tractable cases 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 [15], including dichotomies in the complexity of model checking for circumscription [26], "inverse" satisfiability [24], and computing an assignment maximizing the number of satisfied constraints [25]. 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 [15, 16], which shows that the only tractable cases in this context are QUANTIFIED 2-SAT [1], QUANTIFIED HORN SAT [12], and QUANTIFIED XOR-SAT [15], 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 case of the QCSP that is a candidate for being tractable, one would have the ability to immediately identify whether or not tractability of the case can be deduced from its constraint language; if not, it would follow that other features of the case need to be utilized in a proof of tractability. Indeed, the tractable and intractable cases identified by classification theorems give crisp theoretical results constituting an extremely convenient starting point for performing complexity analysis.

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 cases of the CSP; see, for example, the papers [23, 19, 21, 18, 27, 9, 17, 11, 6, 7]. One 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 [5], which resolved a question from Schaefer's original paper [30] that had been open for over two decades.

Algebra and polymorphisms. A powerful algebraic approach to studying complexity and the relative expressiveness of constraint languages was introduced in [23, 20] and further studied, for example, in [21, 22, 18, 10, 9, 17, 11, 5–7]. 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 *invariant*, called the *polymorphisms* of a constraint language. In particular, sets of polymorphisms are linked to constraint languages (which are sets of relations) via a Galois connection, and two different constraint languages having the same polymorphisms give rise to cases 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 polymorphisms; 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 [5], and can also be used in conjunction with Post's classification theorem [28, 2] to succinctly derive Schaefer's theorem: see, for example, [2].

Quantified constraint satisfaction. Recently, the issue of QCSP complexity based on constraint languages in domains of size greater than two was studied by Börner, Bulatov, Krokhin and Jeavons [3] and the present author [13, 14]. Both of these works used the algebraic approach 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 [13] was general technology for demonstrating the tractability of constraint languages, while [14] studied the complexity of 2-semilattice polymorphisms.

1.2 Contributions of this paper

In this paper, we study QCSP complexity by adopting the approach used to study CSP complexity in [9, 11, 4, 8]: we seek the most general tractability results possible by focusing on *maximal constraint languages*. Maximal constraint languages are those constraint languages that can express any constraint with the help of any relation not contained in the language. Because a constraint language that can express all relations is intractable, maximal constraint languages are the largest constraint languages that could possibly be tractable. The investigation of such languages has played a key role in understanding tractability in the

CSP setting: all of the tractability results identified by Schaefer's theorem apply to maximal constraint languages, and the investigation of maximal constraint languages in domains of size larger than two has resulted in the identification of new tractable cases of the CSP [9, 11, 4, 8].

Classification of maximal constraint languages. We give a *full classification theorem* on maximal constraint languages, showing that each gives rise to a case of the QCSP that is either polynomial-time decidable, or intractable (by which we mean NP-hard or coNP-hard). In order to obtain this theorem, we make use of a theorem of Rosenberg [29] from universal algebra which yields an algebraic description of maximal constraint languages, showing that any maximal constraint language has a polymorphism of one of five types. Most of the effort in obtaining our classification theorem is concentrated in studying one of the five types of maximal constraint languages, namely, those having a binary idempotent polymorphism. We remark that in the CSP setting, a classification of maximal constraint languages was only recently obtained [8], and there the difficult case was also this particular type.

Binary idempotent polymorphisms. Intriguingly, we show that maximal constraint languages invariant under a binary idempotent polymorphism give rise to *four* modes of behavior in the context of quantified constraint satisfaction. In our study, we consider such binary polymorphisms in two classes, those that act as a projection on some two-element domain, and those that do not. Those that do act as a projection give rise to cases of the QCSP that are either NP-complete or PSPACE-complete. Those that do not act as a projection can be assumed to be commutative, and give rise to cases of the QCSP that are either in P or coNP-hard; our demonstration of this fact generalizes the main result of [14]. We leave the exact complexity analysis of the coNP-hard cases as a fascinating issue for future research; we conjecture that these cases are contained in coNP, and are hence coNP-complete.

The fact that the binary polymorphisms investigated here fall into four different regimes of complexity can be contrasted with complexity results on the CSP, where all constraint languages that have been studied have been shown to be either in P or NP-complete. We believe that our results give evidence that, relative to study of the CSP, study of the QCSP is likely to require the utilization of a greater diversity of techniques, and be much richer from a complexity-theoretic standpoint.

Symmetric polymorphisms and set functions. Our study of commutative binary polymorphisms is in fact carried out in a more general setting, that of idempotent symmetric polymorphisms. We *fully classify* idempotent symmetric polymorphisms, showing that for any such polymorphism, the corresponding case of the QCSP is either reducible to its CSP restriction or is coNP-hard, and that an algebraic criterion determines which of the two cases holds (Theorem 17).

Significantly, we show that the ideas used to study symmetric polymorphisms can be deployed to give a *full classification* of idempotent *set functions* in the

QCSP. Set functions have been studied in CSP complexity [18] and guarantee tractability in the CSP setting. As with symmetric polymorphisms, we show that set functions beget cases of the QCSP that are either P or coNP-hard. This classification result resolves an open question naturally arising from [18], namely, to take the class of problems shown therein to be tractable in the CSP, and to give a complexity classification of these problems in the QCSP.

Algebra and complexity. We believe our results to be beautiful examples of the fascinating interplay between algebra and complexity taking place in the setting of constraint satisfaction–an interplay that we feel to be deserving of more attention.

We remark that much of our study concerns *idempotent* polymorphisms. Idempotent polymorphisms give rise to constraint languages containing all *constant relations*, by which we mean arity one relations of size one. Such constraint languages have the desirable robustness property that for any (QCSP or CSP) instance over the constraint language, when a variable is instantiated with a value, the resulting instance is still over the constraint language.

2 Preliminaries

We use the notation $[n]$ to denote the set containing the first n positive integers, $\{1, \ldots, n\}$. We use t_i to denote the ith coordinate of a tuple $\overline{t}$.

2.1 Quantified constraint satisfaction

We now set the basic terminology of quantified constraint satisfaction to be used. Our definitions and notation are fairly standard, and similar to those used in other papers on (quantified) constraint satisfaction. Throughout, we use D to denote a domain, which here is a nonempty set of finite size.

Definition 1. *A* relation *(over D) is a subset of D^k. A* constraint *(over D) is an expression of the form $R(\overline{w})$, where R is a relation over D and $\overline{w}$ is a tuple of variables with the same arity as R. A* constraint language *is a set of relations, all of which are over the same domain.*

Intuitively, a constraint restricts the permissible values that can be given to a set of variables; the variable tuple specifies the variables that are restricted, while the corresponding relation specifies the values that the variable tuple may take on. Formally, we consider an arity k constraint $R(w_1, \ldots, w_k)$ to be *satisfied* under an interpretation f defined on the variables $\{w_1, \ldots, w_k\}$ if $(f(w_1), \ldots, f(w_k)) \in R$.

Definition 2. *A* quantified formula *is an expression of the form $Q_1 v_1 \ldots Q_n v_n \mathcal{C}$ where for all $i \in [n]$, Q_i is a quantifier from the set $\{\forall, \exists\}$ and v_i is a variable; and, $\mathcal{C}$ is a* constraint network*, that is, a finite set of constraints over the same domain, with variables from $\{v_1, \ldots, v_n\}$. A quantified formula is said to be over a constraint language Γ if every constraint in its constraint network $\mathcal{C}$ has relation from Γ.*

Note that, in this paper, we only consider quantified formulas without free variables. Truth of a quantified formula is defined just as in first-order logic; the constraint network $\mathcal{C}$ is interpreted as the conjunction of constraints it contains. The QCSP is the problem of deciding, given a quantified formula, whether or not it is true; the CSP can be defined as the restricted version of the QCSP where all quantifiers appearing must be existential. We are interested in the following parameterized version of the QCSP.

Definition 3. *Let Γ be a constraint language. The* $\mathsf{QCSP}(\Gamma)$ *decision problem is to decide, given as input a quantified formula over Γ, whether or not it is true. We define the* $\mathsf{CSP}(\Gamma)$ *decision problem as the restriction of* $\mathsf{QCSP}(\Gamma)$ *to instances having only existential quantifiers.*

The present paper is a contribution to the long-term research goal of classifying the complexity of $\mathsf{QCSP}(\Gamma)$ for *all* constraint languages Γ.

2.2 Expressibility and polymorphisms

We now explain how the set of relations expressible by a constraint language, and the polymorphisms of a constraint language, characterize the complexity of the constraint language. Our presentation is based on the papers [23, 20], to which we refer the reader for more information.

Definition 4. *(see [20] for details) When Γ is a constraint language over D, define $\langle\Gamma\rangle$, the set of relations* expressible *by Γ, to be the smallest set of relations containing $\Gamma \cup \{=_D\}$ and closed under permutation, extension, truncation, and intersection. (Here, $=_D$ denotes the equality relation on D.)*

The more relations that a constraint language Γ can express, the higher in complexity it is.

Proposition 5. *Let Γ_1, Γ_2 be constraint languages where Γ_1 is finite. If $\langle\Gamma_1\rangle \subseteq \langle\Gamma_2\rangle$, then* $\mathsf{QCSP}(\Gamma_1)$ *reduces to* $\mathsf{QCSP}(\Gamma_2)$.[1]

From Proposition 5, we can see that two finite constraint languages that express exactly the same relations are reducible to one another, and hence of the same complexity. (Up to certain technicalities that are not essential for understanding the new results of this paper, all of our discussion also holds for the case of infinite constraint languages.)

We now introduce the notion of polymorphism. An operation μ of rank k is a polymorphism of a relation R if, for any choice of k tuples $\overline{t_1}, \ldots, \overline{t_k}$ from R, the tuple obtained by acting on the tuples $\overline{t_i}$ in a coordinate-wise manner by μ, is also contained in R.

[1] Note that the only form of reduction we consider in this paper is many-one polynomial-time reduction.

Definition 6. *An operation $\mu : D^k \to D$ is a* polymorphism *of a relation $R \subseteq D^m$ if for all tuples $\overline{t_1}, \ldots, \overline{t_k} \in R$, the tuple $(\mu(t_{11}, \ldots, t_{k1}), \ldots, \mu(t_{1m}, \ldots, t_{km}))$ is in R. An operation μ is a polymorphism of a constraint language Γ if μ is a polymorphism of all relations $R \in \Gamma$. When μ is a polymorphism of R (Γ), we also say that R (Γ) is* invariant *under μ.*

We will be interested in the set of all polymorphisms of a constraint language Γ, as well as the set of all relations invariant under all operations in a given set.

Definition 7. *Let $\mathcal{O}_D$ denote the set of all finite rank operations over D, and let $\mathcal{R}_D$ denote the set of all finite arity relations over D.*

When $\Gamma \subseteq \mathcal{R}_D$ is a set of relations (that is, a constraint language), we define

$$\mathsf{Pol}(\Gamma) = \{\mu \in \mathcal{O}_D \mid \mu \text{ is a polymorphism of } \Gamma\}.$$

When $F \subseteq \mathcal{O}_D$ is a set of operations, we define

$$\mathsf{Inv}(F) = \{R \in \mathcal{R}_D \mid R \text{ is invariant under all operations } \mu \in F\}.$$

When f is a single operation, we use $\mathsf{Inv}(f)$ as notation for $\mathsf{Inv}(\{f\})$, and $\mathsf{QCSP}(f)$ ($\mathsf{CSP}(f)$) as notation for $\mathsf{QCSP}(\mathsf{Inv}(f))$ ($\mathsf{CSP}(\mathsf{Inv}(f))$).

Theorem 8. *For any constraint language Γ, it holds that $\langle\Gamma\rangle = \mathsf{Inv}(\mathsf{Pol}(\Gamma))$.*

From Theorem 8, we see that the complexity of a constraint language depends only on its polymorphisms, since its polymorphisms determine the set of relations that it can express, which as we have discussed, characterizes its complexity.

Proposition 9. *Let Γ_1, Γ_2 be constraint languages where Γ_1 is finite.*
If $\mathsf{Pol}(\Gamma_2) \subseteq \mathsf{Pol}(\Gamma_1)$, then $\mathsf{QCSP}(\Gamma_1)$ reduces to $\mathsf{QCSP}(\Gamma_2)$. Moreover, if both Γ_1 and Γ_2 are finite and $\mathsf{Pol}(\Gamma_1) = \mathsf{Pol}(\Gamma_2)$, then $\mathsf{QCSP}(\Gamma_1)$ and $\mathsf{QCSP}(\Gamma_2)$ are equivalent in that they reduce to one another.

3 Maximal constraint languages

We study the most general forms of constraint language restrictions by considering *maximal constraint languages*, the largest possible constraint languages that cannot express all relations.

Definition 10. *A constraint language Γ is* maximal *if $\langle\Gamma\rangle$ is not the set of all relations, but for any relation R not contained in Γ, $\langle\Gamma \cup \{R\}\rangle$ is the set of all relations.*

A theorem of Rosenberg [29] demonstrates that all maximal constraint languages are invariant under an operation of one of five particular forms. In order to state this theorem, we require some new terminology. An operation $f : D^k \to D$ is a *majority* operation if $k = 3$ and for all $a, a' \in D$ 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 $a_1, a_2, a_3 \in D$ 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 $(D, *, ^{-1})$ is an Abelian group; a *projection* if there exists $i \in [k]$ such that for all $a_1, \ldots, a_k \in D$, $f(a_1, \ldots, a_k) = a_i$; and a *semiprojection* if $k \geq 3$, f is not a projection, and there exists $i \in [k]$ such that for all $a_1, \ldots, a_k \in D$, $|\{a_1, \ldots, a_k\}| < k$ implies $f(a_1, \ldots, a_k) = a_i$.

Theorem 11. *(follows from [29]) If Γ is a maximal constraint language, then $\Gamma = \mathsf{Inv}(f)$ for an operation f having one of the following five types: a unary operation which is either a bijection or acts identically on its range, a semiprojection, a majority operation, an affine operation, a binary idempotent operation that is not a projection.*

The first two types of operations in Theorem 11 give rise to hard cases of the $\mathsf{QCSP}(\Gamma)$ problem.

Theorem 12. *Let Γ be a maximal constraint language of the form $\mathsf{Inv}(f)$, where f is a unary operation which is either a bijection or acts identically on its range. Then $\mathsf{QCSP}(\Gamma)$ is PSPACE-complete.*

Theorem 13. *[11] Let Γ be a maximal constraint language of the form $\mathsf{Inv}(f)$, where f is a semiprojection. Then $\mathsf{QCSP}(\Gamma)$ is NP-hard.*

The next two types of operations in Theorem 11 have previously been shown to be tractable.

Theorem 14. *[13, 3] Let Γ be a constraint language invariant under a majority operation or invariant under an affine operation. Then $\mathsf{QCSP}(\Gamma)$ is in P.*

The following is the statement of our classification theorem on maximal constraint languages.

Theorem 15. *(Classification of maximal constraint languages) Let Γ be a maximal constraint language. One of the following five conditions holds:*

- *$\mathsf{QCSP}(\Gamma)$ is in P.*
- *$\mathsf{QCSP}(\Gamma)$ is coNP-hard and $\Gamma = \mathsf{Inv}(f)$ for f a binary commutative idempotent operation that is not a projection.*
- *$\mathsf{QCSP}(\Gamma)$ is NP-complete.*
- *$\mathsf{QCSP}(\Gamma)$ is NP-hard and $\Gamma = \mathsf{Inv}(f)$ for f a semiprojection.*
- *$\mathsf{QCSP}(\Gamma)$ is PSPACE-complete.*

Proof. For the first four types of maximal constraint languages in Theorem 11, the result holds by Theorems 12, 13, and 14. Otherwise, consider a maximal constraint language $\Gamma = \mathsf{Inv}(f)$ for f a binary idempotent operation that is not a projection. Using a technique similar to that of the proof of [11, Lemma 3], it can be shown that $\Gamma = \mathsf{Inv}(f')$ for f' a binary idempotent operation such that, for every two-element subset $\{a, b\}$ of D, either $f'(a, b) = f'(b, a)$, or f' acts as a projection on $\{a, b\}$.

– If there is no two-element subset $\{a, b\}$ such that f' acts as a projection on $\{a, b\}$, the function f' is commutative and $\mathsf{QCSP}(\Gamma)$ either reduces to $\mathsf{CSP}(\Gamma)$ or is coNP-hard by Theorem 17 (Section 4). In the former case, $\mathsf{CSP}(\Gamma)$, and hence $\mathsf{QCSP}(\Gamma)$, is in P by [8, Theorem 5].
– Otherwise, there exists at least one two-element subset $\{a, b\}$ such that f' acts as a projection on $\{a, b\}$, and $\mathsf{QCSP}(\Gamma)$ is either NP-complete or PSPACE-complete by Theorem 22 (Section 5).

□

4 Symmetric polymorphisms

In this section, we develop algebraic theory that permits us to present a complete classification of idempotent symmetric polymorphisms in the QCSP, as well as a complete classification of idempotent set functions. We say that a polymorphism (or function) $f : D^k \to D$ is *symmetric* if for all $a_1, \ldots, a_k \in D$ and for all permutations $\pi : [k] \to [k]$, it holds that $f(a_1, \ldots, a_k) = f(a_{\pi(1)}, \ldots, a_{\pi(k)})$. Note that in this section, we consider symmetric polymorphisms of all arities, and not just binary polymorphisms.

Let $f : D^k \to D$ be an idempotent symmetric function. We say that an element $a \in D$ can *f-hit* an element $b \in D$ in one step if there exist elements $a_1, \ldots, a_{k-1} \in D$ such that $f(a, a_1, \ldots, a_{k-1}) = b$. Let us say that $a \in D$ can f-hit $b \in D$ (in m steps) if there exist elements $d_0, \ldots, d_m \in D$ such that $a = d_0$, $b = d_m$ and for all $i = 0, \ldots, m-1$ it holds that d_i can hit d_{i+1} in one step. Notice that the "can f-hit" relation is reflexive and transitive.

Define a set $C \subseteq D$ to be *coherent* with respect to f if it is nonempty and for all $a_1, \ldots, a_k \in D$, the following holds: if $\{a_1, \ldots, a_k\} \setminus C$ is nonempty, then $f(a_1, \ldots, a_k) \notin C$ (equivalently, if $f(a_1, \ldots, a_k) \in C$, then $a_1, \ldots, a_k \in C$). When $S \subseteq D$ is a nonempty set that is not coherent, if $\{a_1, \ldots, a_k\} \setminus S$ is nonempty and $f(a_1, \ldots, a_k) \in S$, we say that $(a_1, \ldots, a_k)$ is a *witness to the non-coherence of S*.

Observe that the union of any two coherent sets is also a coherent set. In addition, when C_1, C_2 are two coherent sets with a non-trivial intersection, their intersection $C_1 \cap C_2$ is also a coherent set. We use $\langle d \rangle$ to denote the smallest coherent set containing d.

Lemma 16. *All elements of $\langle d \rangle$ can hit d.*

Let us say that a coherent set is *minimal* if it is minimal (among coherent sets) with respect to the subset $\subseteq$ ordering. We show that whether or not an idempotent symmetric function f has a *unique* minimal coherent set in fact determines the complexity of $\mathsf{QCSP}(f)$.

Theorem 17. *Let f be an idempotent symmetric function. If there is a unique minimal coherent set with respect to f, then $\mathsf{QCSP}(f)$ reduces to $\mathsf{CSP}(f)$; otherwise, $\mathsf{QCSP}(f)$ is coNP-hard.*

Similar techniques can be used to establish a classification result on idempotent *set functions*. We consider a *set function* (on domain D) to be a mapping from $\wp(D) \setminus \{\emptyset\}$ to D, where $\wp(D)$ denotes the power set of D. A set function $f : \wp(D) \setminus \{\emptyset\} \to D$ is considered to be *idempotent* if for all $d \in D$, it holds that $f(\{d\}) = d$. We consider a relation R of arity k to be *invariant* under a set function $f : \wp(D) \setminus \{\emptyset\} \to D$ if for any non-empty subset $S \subseteq R$, it holds that the tuple $(f(\{t_1 : \overline{t} \in S\}), \ldots, f(\{t_k : \overline{t} \in S\}))$ is in R. Equivalently, R is invariant under $f : \wp(D) \setminus \{\emptyset\} \to D$ if it is invariant under all of the functions $f_i : D^i \to D$ defined by $f_i(d_1, \ldots, d_i) = f(\{d_1, \ldots, d_i\})$, for $i \geq 1$. Set functions were studied in the context of CSP complexity by Dalmau and Pearson [18]; among other results, they showed that any problem of the form $\mathsf{CSP}(f)$, for f a set function, is polynomial-time decidable.

We can define notions similar to those in the beginning of this section for set functions. In particular, when f is a set function, we say that an element $a \in D$ can *f-hit* an element $b \in D$ in one step if there exists a subset $A \subseteq D$ such that $f(\{a\} \cup A) = b$. We define a set $C \subseteq D$ to be *coherent* with respect to f if it is nonempty and for all nonempty $A \subseteq D$, the following holds: if $A \setminus C$ is nonempty, then $f(A) \notin C$. Using these notions, it is possible to give proofs analogous to those for the previous results of this section, yielding the following classification theorem.

Theorem 18. *Let $f : \wp(D) \setminus \{\emptyset\} \to D$ be an idempotent set function. If there is a unique minimal coherent set with respect to f, then $\mathsf{QCSP}(f)$ reduces to $\mathsf{CSP}(f)$ (and is hence in P by [18]); otherwise, $\mathsf{QCSP}(f)$ is coNP-hard.*

5 Commutative-projective operations

As we have indicated (proof of Theorem 15), all binary operations giving rise to maximal constraint languages can be seen as having one of two types. The previous section was concerned with a generalization of the first type, *commutative* binary operations; this section studies the second type, *commutative-projective* operations.

Definition 19. *A* commutative-projective *operation is a binary idempotent operation $f : D^2 \to D$ such that for every two-element subset $\{a, b\}$ of D, either $f(a, b) = f(b, a)$, or f acts as a projection on $\{a, b\}$; and, there exists a two-element subset $\{a, b\}$ of D such that f acts as a projection on $\{a, b\}$.*

Fix a commutative-projective operation $f : D^2 \to D$. Based on f, we define two directed graphs G_1, G_2 as follows. Both of these graphs have vertex set D. Let there be an edge from a to b in both G_1 and G_2 if there exists $d \in D$ such that $f(a, d) = f(d, a) = b$. In addition, let there be an edge from a to b as well as an edge from b to a in G_i if on the two-element set $\{a, b\}$ the operation f acts as the projection onto the ith coordinate. We have the following results.

Lemma 20. *Suppose that there exists $d_0 \in D$ such that for both $i \in \{1, 2\}$ and for every $d \in D$, there is a path from d_0 to d in G_i. Then, $\mathsf{QCSP}(f)$ is NP-complete.*

For each $i \in \{1,2\}$, define $\leq_i$ to be the partial ordering on strongly connected components of G_i where $C \leq_i C'$ if there exist vertices $v \in C, v' \in C'$ such that there is a path from v to v' in G_i. We define a component C to be minimal in G_i if for all components C' such that $C' \leq_i C$, it holds that $C' = C$.

Lemma 21. *Suppose that one (or both) of the graphs G_1, G_2 has more than one minimal component. Then,* $\mathsf{QCSP}(f)$ *is PSPACE-complete.*

Theorem 22. *Let $f : D^2 \to D$ be a commutative-projective operation such that* $\mathsf{Inv}(f)$ *is a maximal constraint language. The problem* $\mathsf{QCSP}(f)$ *is either NP-complete or PSPACE-complete.*

Acknowledgements. The author thanks the anonymous referees for their helpful 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, N. Creignou, S. Reith, and H. Vollmer. Playing with boolean blocks, part II: constraint satisfaction problems. *ACM SIGACT-Newsletter*, 35(1):22–35, 2004.
3. F. Börner, A. Bulatov, A. Krokhin, and P. Jeavons. Quantified constraints: Algorithms and complexity. In *Computer Science Logic 2003*, 2003.
4. Andrei Bulatov. Combinatorial problems raised from 2-semilattices. Manuscript.
5. Andrei 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.
6. Andrei Bulatov. Malt'sev constraints are tractable. Technical Report PRG-RR-02-05, Oxford University, 2002.
7. Andrei Bulatov. Tractable conservative constraint satisfaction problems. In *Proceedings of 18th IEEE Symposium on Logic in Computer Science (LICS '03)*, pages 321–330, 2003.
8. Andrei Bulatov. A graph of a relational structure and constraint satisfaction problems. In *Proceedings of 19th IEEE Annual Symposium on Logic in Computer Science (LICS'04)*, 2004.
9. Andrei Bulatov and Peter Jeavons. Tractable constraints closed under a binary operation. Technical Report PRG-TR-12-00, Oxford University, 2000.
10. Andrei Bulatov, Andrei 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.
11. Andrei Bulatov, Andrei Krokhin, and Peter Jeavons. The complexity of maximal constraint languages. In *ACM Symposium on Theory of Computing*, pages 667–674, 2001.
12. Hans Kleine Büning, Marek Karpinski, and Andreas Flögel. Resolution for quantified boolean formulas. *Information and Computation*, 117(1):12–18, 1995.

13. Hubie Chen. Collapsibility and consistency in quantified constraint satisfaction. In *AAAI*, 2004.
14. Hubie Chen. Quantified constraint satisfaction and 2-semilattice polymorphisms. In *CP*, 2004.
15. 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.
16. 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.
17. Victor Dalmau. A new tractable class of constraint satisfaction problems. In *6th International Symposium on Artificial Intelligence and Mathematics*, 2000.
18. Victor Dalmau and Justin Pearson. Closure functions and width 1 problems. In *CP 1999*, pages 159–173, 1999.
19. 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.
20. Peter Jeavons. On the algebraic structure of combinatorial problems. *Theoretical Computer Science*, 200:185–204, 1998.
21. Peter Jeavons, David Cohen, and Martin Cooper. Constraints, consistency, and closure. *Articial Intelligence*, 101(1-2):251–265, 1998.
22. Peter Jeavons, David Cohen, and Justin Pearson. Constraints and universal algebra. *Annals of Mathematics and Artificial Intelligence*, 24(1-4):51–67, 1998.
23. P.G. Jeavons, D.A. Cohen, and M. Gyssens. Closure properties of constraints. *Journal of the ACM*, 44:527–548, 1997.
24. D. Kavvadias and M. Sideri. The inverse satisfiability problem. *SIAM Journal on Computing*, 28(1):152–163, 1998.
25. 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.
26. L. M. Kirousis and P. G. Kolaitis. The complexity of minimal satisfiability problems. In *Proceedings 18th Annual Symposium on Theoretical Aspects of Computer Science*, volume 2010 of *Lecture Notes in Computer Science*, pages 407–418. Springer-Verlag, 2001.
27. Ph.G. Kolaitis and M.Y. Vardi. Conjunctive-query containment and constraint satisfaction. *Journal of Computer and System Sciences*, 61:302–332, 2000.
28. Emil L. Post. *The Two-Valued Iterative Systems of Mathematical Logic*. Princeton University Press, 1941.
29. 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.
30. Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 216–226, 1978.

A Proof of Lemma 16

Proof. We define a sequence of sets S_i as follows. Define $S_0 = \{d\}$. If S_i is coherent, then S_{i+1} is undefined; otherwise, let $(a_1^i, \ldots, a_k^i)$ be a witness to the non-coherence of S_i, and define $S_{i+1} = S_i \cup \{a_1^i, \ldots, a_k^i\}$. Notice that when S_{i+1}

is defined, any coherent set T containing S_i must contain the elements $a_1^i, \ldots, a_k^i$, and hence S_{i+1} (otherwise, $(a_1^i, \ldots, a_k^i)$ is a witness to the non-coherence of T). Consequently, we have $\langle d \rangle = \cup_i S_i$.

Observe that, by definition, any element of S_{i+1} can hit an element of S_i. It follows that every element of $\langle d \rangle$ can hit d, as desired. □

B Proof of Theorem 17

In order to prove Theorem 17, we make use of the following theorem, which can be proved using ideas in [13].

Theorem 23. *Let f be an idempotent function. Suppose that there exists a function $g : D^m \to D$ contained in the clone generated by f and an element $d \in D$ such that all m of the arity $(m-1)$ functions obtained by instantiating one of the arguments of g with d are surjective. Then* $\mathsf{QCSP}(f)$ *reduces to (and is hence equivalent to)* $\mathsf{CSP}(f)$.

Proof. (Theorem 17 - reduction to CSP) Define $f_1 = f$ and $f_{i+1} : D^{k^{i+1}} \to D$ by $f_{i+1}(\overline{x_1}, \ldots, \overline{x_k}) = f(f_i(\overline{x_1}), \ldots, f_i(\overline{x_k}))$ for all $\overline{x_1}, \ldots, \overline{x_k} \in D^{k^i}$. We show that there exists a constant N and $d \in D$ such that all k^N of the arity $(k^N - 1)$ functions obtained by instantiating one of the arguments of $f_N : D^{k^N} \to D$ with d, are surjective. The theorem then follows from Theorem 23.

Suppose that there is a unique minimal coherent set. Fix b to be any element of the unique minimal coherent set. Every coherent set contains b, so for all $d \in D$, we have $b \in \langle d \rangle$. By Lemma 16, the element b can hit all elements $d \in D$. Let N be a sufficiently high constant so that for all $d \in D$, the element b can hit d in N steps. Define functions $f_i' : D^{i(k-1)} \to D$ by $f_1'(x_1, \ldots, x_{k-1}) = f(b, x_1, \ldots, x_{k-1})$ and $f_{i+1}'(\overline{y}, x_1, \ldots, x_{k-1}) = f(f_i'(\overline{y}), x_1, \ldots, x_{k-1})$. By choice of N, the function f_N' is surjective. By using the symmetry and idempotence of f, it can be shown that the surjectivity of f_N' implies that the constant N and element $b \in D$ satisfies the desired property. □

Proof. (Theorem 17 - coNP-hardness) We demonstrate coNP-hardness by reducing from the propositional tautology problem. Let $F(y_1, \ldots, y_n)$ be an instance of this problem, where F is a circuit with input gates having labels $y_1, \ldots, y_n$ We assume that all non-input gates of F are either AND or NOT gates, and assign all non-input gates labels $x_1, \ldots, x_m$.

Let C_0 and C_1 be distinct (and hence disjoint) minimal coherent sets. Fix c_0 and c_1 to be elements of C_0 and C_1, respectively. Define R as $D \setminus (C_0 \cup C_1)$. Define N to be the arity 2 relation

$$(C_0 \times \{c_1\}) \cup (C_1 \times \{c_0\}) \cup (R \times D)$$

Define A to be the arity 3 relation

$$(C_0 \times C_0 \times \{c_0\}) \cup (C_0 \times C_1 \times \{c_0\}) \cup (C_1 \times C_0 \times \{c_0\}) \cup (C_1 \times C_1 \times \{c_1\}) \cup$$

$$(R \times (C_0 \cup C_1) \times D) \cup ((C_0 \cup C_1) \times R \times D) \cup (R \times R \times D)$$

It is fairly straightforward to verify that the relations N and A have f as polymorphism. Here, we verify that A has f as polymorphism. Suppose that $\overline{s_1}, \ldots, \overline{s_k} \in A$, and set $\overline{t} = f(\overline{s_1}, \ldots, \overline{s_k})$; we wish to show that $\overline{t} \in A$. If one (or both) of t_1, t_2 are contained in R, then $\overline{t} \in A$ because $R \times D \times D \subseteq A$ and $D \times R \times D \subseteq A$. So, suppose that $t_1 \in C_i$ and $t_2 \in C_j$, with $i, j \in \{0, 1\}$. Since C_i and C_j are coherent, we have $s_{11}, \ldots, s_{k1} \in C_i$ and $s_{12}, \ldots, s_{k2} \in C_j$. By definition of A, it follows that $s_{13}, \ldots, s_{k3} = c_{i \wedge j}$; by the idempotence of f, this implies $t_3 = c_{i \wedge j}$, from which it follows that $\overline{t} \in A$.

Based on the tautology instance F, we create a quantified formula with quantifier prefix

$$\forall y_1 \ldots \forall y_n \exists x_1 \ldots \exists x_m$$

and with constraint network 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 constraint $A(v, v', x_i)$.
- For each NOT gate x_i with input $v \in \{y_1, \ldots, y_n\} \cup \{x_1, \ldots, x_m\}$, include the constraint $N(v, x_i)$.
- For the output gate x_{output}, include the constraint $(D \setminus C_0)(x_{\mathsf{output}})$.

We verify the reduction to be correct as follows. Observe that the constraint network is satisfiable under all assignments $f : \{y_1, \ldots, y_n\} \to D$ to the universally quantified variables if and only if it is satisfiable under all c_i-assignments $f : \{y_1, \ldots, y_n\} \to \{c_0, c_1\}$. This is because when $f : \{y_1, \ldots, y_n\} \to D$ is any assignment, for any assignment $f' : \{y_1, \ldots, y_n\} \to \{c_0, c_1\}$ such that $f(y_i) \in C_0 \cup C_1 \Rightarrow [\{f(y_i), f'(y_i)\} \subseteq C_0$ or $\{f(y_i), f'(y_i)\} \subseteq C_1]$ the constraint network on $\{x_1, \ldots, x_m\}$ obtained by instantiating the universal variables under f' is at least as constrained than the constraint network on $\{x_1, \ldots, x_m\}$ obtained by instantiating the universal variables under f.

Let $g : \{y_1, \ldots, y_n\} \to \{0, 1\}$ be any assignment to the input gates of F. It is straightforward to verify that, under the mapping $g' : \{y_1, \ldots, y_n\} \to \{c_0, c_1\}$ defined by $g'(y_i) = c_{g(y_i)}$, the only assignment to the variables x_i satisfying the first two types of constraints is the assignment taking x_i to c_0 if the gate x_i of F is equal to 0 under input g, and c_1 if the gate x_i of F is equal to 1 under input g. Because of the third type of constraint (on the output gate), the circuit F is true under g if and only if the constraint network is satisfiable under g', and hence the circuit F is a tautology if and only if the quantified formula is true. □

C Proof of Lemma 20

Proof. There exists a two-element subset $B \subseteq D$ such that f acts as a projection on B; any relation over B is invariant under f. Consequently, the problem $\mathsf{CSP}(f)$ is NP-complete, and the problem $\mathsf{QCSP}(f)$ is NP-hard. We need to show that $\mathsf{QCSP}(f)$ is in NP.

Define functions $f_i : D^{2^i} \to D$, for $i \geq 1$, as in the proof of Theorem 17. As in that proof, we show that there exists a constant N such that all 2^N of the arity $(2^N - 1)$ functions obtained by instantiating one of the arguments of $f_N : D^{2^N} \to D$ with d_0, are surjective. The theorem then follows from Theorem 23, since then $\mathsf{QCSP}(f)$ reduces to $\mathsf{CSP}(f)$, which is in NP.

For $i \in \{1, 2\}$, pick M_i to be a sufficiently high integer so that for any vertex $d \in D$, there is a path from d_0 to d in G_i of length less than or equal to M_i. We claim that choosing N as $M_1 + M_2 - 1$ suffices, and now explain why. First, consider instantiating the first argument of $f_j : D^{2^j} \to D$ with d_0. The resulting operation can be visualized as a binary tree of height j where the interior nodes have label f, the root node is on top, the first (leftmost) leaf has value d_0, and the remaining $2^j - 1$ leaves are the arguments. It can be seen that all elements of D that are within distance j of d_0 in G_2 are contained in the image of the resulting arity $2^j - 1$ function. Next, consider instantiating the the last argument of $f_j : D^{2^j} \to D$ with d_0. In this case, all elements of D that are within distance j of d_0 in G_1 are contained in the image of the resulting arity $2^j - 1$ function. In the general case where an arbitrary argument of f_N is instantiated with d_0, choosing N as $M_1 + M_2 - 1$ ensures that all elements of D are contained in the image of the resulting arity $2^j - 1$ function, because in traversing a binary tree of height $M_1 + M_2 - 1$ from the leaf level to the root, one makes at least M_2 moves to the right or at least M_1 moves to the left. □

D Proof of Lemma 21

Proof. We assume without loss of generality that C_1, C_2 are disjoint minimal components of G_1. If there exists a two-element subset $S \subseteq D$ on which f acts as projection onto the second coordinate, define a_1, a_2 to be the elements of S. Otherwise, there exists a two-element subset $S \subseteq D$ on which f acts as projection onto the first coordinate; let a_1, a_2 denote the elements of S.

Define R to be $D \setminus (C_1 \cup C_2)$, and define T to be the arity 2 relation $T = (C_1 \times \{a_1\}) \cup (C_2 \times \{a_2\}) \cup (R \times \{a_1, a_2\})$. It can be verified that the relation T is invariant under f. A key fact is that when $c, c' \in D$ and c, c' do not lie in the same set (of the sets C_1, C_2, and R), either $f(c, c') = f(c', c) \in R$ or f acts as a projection on $\{c, c'\}$; in the latter case, f must act on $\{c, c'\}$ as projection onto the second coordinate, and in this case it also acts as projection onto the second coordinate on $\{a_1, a_2\}$, by choice of a_1, a_2.

We show that $\mathsf{QCSP}(f)$ is PSPACE-hard by reduction from an arbitrary instance ϕ of the QCSP over the two-element domain $\{a_1, a_2\}$. We create an instance ϕ' of $\mathsf{QCSP}(f)$ based on ϕ as follows. For each universally quantified variable y of ϕ, the formula ϕ' contains two variables: u_y, which is universally quantified, and v_y, which is existentially quantified. The quantifier prefix of ϕ' is obtained by substituting, in the quantifier prefix of ϕ, each universally quantified variable $\forall y$ with $\forall u_y \exists v_y$. The constraint network of ϕ' is obtained from the constraint network of ϕ by copying all constraints but replacing each occurrence of y with v_y, and adding, for each universally quantified variable y in ϕ, a con-

straint $T(u_y, v_y)$. The key feature of T that we use here is that in a constraint of the form $T(u_y, v_y)$, setting u_y to an element of C_1 (respectively, C_2) forces v_y to take on the value a_1 (respectively, a_2). □

E Proof of Theorem 22

Proof. Suppose that f acts as a projection on all two-element domains. Then the edge sets of G_1, G_2 are symmetric, and it is straightforward to verify that one of Lemmas 20, 21 apply: if Lemma 21 does not apply, then both G_1 and G_2 are strongly connected and any element of D can be picked as d_0 to apply Lemma 20.

Suppose that f does not act as a projection on all two-element domains. Define $g : D^2 \to D$ by $g(x_1, x_2) = f(f(x_1, x_2), f(x_2, x_1))$. For all elements $a, b \in D$, if $f(a, b) = f(b, a)$ then $g(a, b) = g(b, a)$; and, if f acts as a projection on $\{a, b\}$, then g acts as projection onto the first coordinate on $\{a, b\}$. Thus g is commutative-projective; let G_1, G_2 be the graphs corresponding to g. We have that the edges in G_1 are a superset of the edges in G_2, and that for any edge (a, b) in G_1 but not in G_2, the edge (b, a) is also in G_1 but not in G_2.

We know that $f(a, b) = f(b, a)$ for some distinct elements $a, b \in D$; since $g(a, b) = g(b, a)$, g is not a projection. Since g is a polymorphism of $\mathsf{Inv}(f)$, by the maximality of $\mathsf{Inv}(f)$, we have $\mathsf{Inv}(f) = \mathsf{Inv}(g)$.

If Lemma 21 does not apply to G_1, G_2, then let d_0 be any element of the unique minimal component of G_2. The element d_0 must also be an element of the unique minimal component of G_1, and hence Lemma 20 applies. □

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'