The Complexity of Constraint Satisfaction Games and QCSP

Ferdinand Börner
Institut für Informatik
University of Potsdam
Potsdam, D-14482, Germany
fboerner@rz.uni-potsdam.de

Andrei Bulatov
School of Computing Science
Simon Fraser University
Burnaby BC, Canada V5A 1S6
abulatov@cs.sfu.ca

Hubie Chen
Department of Technology
University Pompeu Fabra
Barcelona 08003, Spain
hubie.chen@upf.edu

Peter Jeavons
Computing Laboratory
University of Oxford
Oxford, OX1 3QD, UK
peter.jeavons@comlab.ox.ac.uk

Andrei Krokhin
Department of Computer Science
Durham University
Durham, DH1 3LE, UK
andrei.krokhin@durham.ac.uk

Abstract

We study the complexity of two-person constraint satisfaction games. An instance of such a game is given by a collection of constraints on overlapping sets of variables, and the two players alternately make moves assigning values from a finite domain to the variables in a specified order. The first player tries to satisfy all constraints, while the other tries to break at least one constraint; the goal is to decide whether the first player has a winning strategy. We show that such games can be conveniently represented by a logical form of quantified constraint satisfaction, where an instance is given by a first-order sentence in which quantifiers alternate and the quantifier-free part is a conjunction of atomic formulas; the goal is to decide whether the sentence is true.

While the problem of deciding such a game is **PSPACE**-complete in general, by restricting the set of allowed constraint predicates, one can obtain infinite classes of constraint satisfaction games of lower complexity. We use the quantified constraint satisfaction framework to study how the complexity of deciding such a game depends on the parameter set of allowed predicates. With every predicate, one can associate certain predicate-preserving operations, called polymorphisms. We show that the complexity of our games is determined by the surjective polymorphisms of the constraint predicates. We illustrate how this result can be used by identifying the complexity of a wide variety of constraint satisfaction games.

Keywords: constraint satisfaction, games, quantified constraint satisfaction problem, polymorphisms, complexity, algorithms.

1 Introduction

The *constraint satisfaction problem* (CSP) provides a general framework in which a wide variety of combinatorial search problems can be expressed in a natural way [18, 23]. An instance of the CSP can be viewed as a collection of predicates on overlapping sets of variables; the aim is to determine whether there exist values for all of the variables such that all of the specified predicates hold simultaneously. Although the CSP, in its general formulation, is **NP**-complete and hence likely to be intractable, it can be parameterized by restricting the set of allowed predicates which can be used as constraints. The problem of classifying the complexity of the CSP (and its many variants) for all possible parameter sets has attracted much attention, because constraint satisfaction problems play an important role in many areas of computer science and artificial intelligence [23]. An important outcome of research in this direction has been the design of sophisticated new polynomial-time algorithms for solving a wide variety of problems (see, for example, [7, 21]). In addition, classification results for the CSP are significant from a complexity-theoretic standpoint, as they provide large subclasses of complexity classes that avoid intermediate complexity. For example, in the case of the class **NP**, a number of dichotomy results have been obtained [6, 7, 18, 26].

The complexity of combinatorial games is also a major line of research (see [27, 28, 41, 49]). In this paper we study the complexity of two-person constraint satisfaction games, in which, given a CSP instance, two players (call them $\exists$ and $\forall$) alternately assign values to the variables in a specified order. Player $\exists$ tries to satisfy all constraints, while player $\forall$ (the adversary) tries to break at least one constraint; the goal is to decide whether player $\exists$ has a winning strategy. Note that the order of variables is specified in every instance, since otherwise (if the players were allowed to choose the variable ordering) the adversary would be able to break constraints too easily. The complexity of some related games was studied in [2, 47]. A different kind of game has already been studied in the context of constraint satisfaction [38] where they were used to prove tractability of certain CSPs.

The CSP can be expressed as the problem of deciding the truth of a given first-order sentence consisting of a conjunction of predicates, where all of the variables are existentially quantified. Hence the CSP generalises the standard propositional *satisfiability* problem, by allowing the possible values for the variables to be chosen from an arbitrary finite set, and allowing the constraints to be arbitrary predicates rather than just clauses. Satisfiability games of the form described above can be conveniently cast as (and are equivalent to) *quantified* satisfiability problems, known as QSAT. Similarly, games of this form on CSP instances are equivalent to *quantified constraint satisfaction problems* (QCSP), in which universal quantifiers are allowed in the sentence, in addition to existential quantifiers [18, 19]. The existentially quantified variables correspond to the moves of $\exists$, and the universally quantified ones to the moves of $\forall$; the (specified) order of moves corresponds to the order of quantifiers in the formula. Note that if the quantifiers do not alternate in a formula then the standard trick is to insert into the prefix appropriately quantified "dummy" variables that do not appear in the quantifier-free part; obviously, this does not affect validity of the formula, and the size of the formula increases by an at most constant factor.

The QCSP framework is actively studied in artificial intelligence, where it is used to model problems, for example, in non-monotonic reasoning [24] and in planning [46]. One motivating example for the study of the QCSP arises in the development of automated systems with certain integrity constraints; such a system should be able to respond to any action of the user (who may be thought of as an adversary) in such a way that the integrity constraints are satisfied. Checking whether or not such a system is safe amounts to solving a QCSP. Several general (superpolynomial or incomplete) algorithms for the Boolean QCSP (that is, QSAT) have been suggested [13, 31, 36, 51], and recently researchers have begun to look for ways to solve non-Boolean QCSP problems [3, 30, 40, 51].

It is not hard to see that QCSP is **PSPACE**-complete in general. However, with certain restrictions on the type of predicates allowed in instances, the constraint satisfaction game may be easier to decide. Our ultimate goal is to determine how the complexity of deciding a constraint satisfaction game depends on the parameter set (of predicates allowed in instances).

The Boolean QCSP and some of its restrictions, such as Quantified 3-SAT, have always been standard examples of **PSPACE**-complete problems [29, 41, 48]. However, for some parameter sets, Boolean QCSP has been shown to be tractable: for all binary predicates in [1], and for Horn predicates in [36]. Indeed, a complete classification for the Boolean QCSP was obtained in [18, 19] (see Theorem 3.8, below).

However, the general non-Boolean QCSP has not yet been systematically studied from the viewpoint of complexity classification. This paper initiates a systematic approach to the complexity classification of the QCSP over arbitrary finite domains. Within the three years between the announcement of (some of) the results of this paper [4, 5, 15] and finishing this extended version, several further papers following this line of research have appeared, e.g., [14, 16, 17, 25].

Obtaining complexity classifications for non-Boolean CSPs is significantly more difficult than for Boolean CSPs: the direct combinatorial approach used in the Boolean case is infeasible, so more involved techniques are required. A far-reaching approach for tackling this general case via graph theory, logic and games has been developed in [20, 26, 37]. However, the most successful approach to date has been the algebraic approach developed in [12, 33, 35] (see also [39]). This approach has led to many new tractability and classification results (see for example, [8, 9, 10, 11, 20, 21, 39]) and, thus far, has culminated in a complete classification of the complexity of CSPs for the three-valued case [6], and the case when all unary predicates are available [7].

In this paper, we extend the algebraic framework that has been used to study the CSP, and we show that certain algebraic objects (surjective polymorphisms) determine the complexity of the QCSP for any given choice of parameter set. We then use this approach to identify several classes of parameter sets yielding a tractable QCSP. The CSP for each of these classes is already known to be tractable [35, 39], but establishing the tractability of the QCSP for these classes requires considerable further effort. Moreover, we show that some surjective polymorphisms that are known to guarantee the tractability of the CSP fail to do so for the QCSP. We also apply the results to classify the complexity of a range of constraint satisfaction games.

The paper is organised as follows. In Section 2, we give the basic definitions, explain the connection between the QCSP and constraint satisfaction games, and provide some examples. Then, in Section 3.1, we outline the algebraic approach to the CSP and, in Section 3.2, we cite known complexity results on the QCSP. An algebraic approach to the QCSP is developed in Section 3.3. Then, in Sections 4.1 and 4.2, we prove the tractability of QCSPs corresponding to Mal'tsev and near-unanimity polymorphisms. Section 5 is devoted to the main intractability result. In Section 6, we show that certain semilattice polymorphisms guarantee tractability of the corresponding QCSPs, while all other semilattice polymorphisms do not. Finally, in Section 7 we obtain a complete classification for QCSPs in which the graphs of all the permutations of the values are available; the result is a 'trichotomy', that is, every problem either belongs to **PTIME**, or is **NP**-complete, or is **PSPACE**-complete.

2 Definitions and Examples

Throughout this paper, we use the standard correspondence between predicates and relations: a relation consists of all tuples of values for which the corresponding predicate holds. We will use the same symbol for a predicate and its corresponding relation, since the meaning will always be clear from the context. We will use $R_D^{(m)}$ to denote the set of all m-ary relations (or predicates) over a set D, and R_D to denote the set of *all* relations over a set D, that is, $R_D = \bigcup_{m=1}^{\infty} R_D^{(m)}$.

The constraint satisfaction problem can be defined as follows.

Definition 2.1 *An instance of the* CSP *on D is a formula $\psi = \psi_1 \wedge \ldots \wedge \psi_q$ where each ψ_i is an atomic formula involving a predicate from R_D. The question is whether ψ is satisfiable.*

An instance *of the* QCSP *is a first-order sentence $\exists v_1 \forall v_2 \ldots \mathcal{Q}_l v_l \ \psi$, where ψ is an instance of the CSP whose variables are chosen from $v_1, \ldots, v_l$ and the quantifiers alternate; the* question *is whether the sentence is true.*

The predicates appearing in an instance will be referred to as *constraints*, since each of them restricts the possible models for ψ in some way.

As explained in the introduction, the version of the QCSP where the quantifiers are not required to alternate is the same from a complexity point of view, since the alternation can always be achieved by using dummy variables. This more general version will often be used in our technical results. Note that by using dummy variables we can also change whether the first (and/or last) quantifier is existential or universal.

Note that the CSP decision problem is the particular case of the QCSP problem where all of the universally quantified variables are dummy.

Since the QCSP is our model for constraint satisfaction games, we will also use the following game-theoretic characterization of QCSP instances.

Definition 2.2 *Let $\phi = \forall y_1 \exists x_1 \ldots \forall y_n \exists x_n \ \psi$ be a QCSP instance over a domain D. A* strategy *for $\exists$ in ϕ is a sequence of mappings $\{\tau_i : D^i \to D\}_{i=1,\ldots,n}$; it is said to be a* winning strategy *if, for any mapping $\sigma : \{y_1, \ldots, y_n\} \to D$ defined on the universally quantified variables of ϕ, the formula ψ is true under the mapping taking each y_i to $\sigma(y_i)$ and each x_i to $\tau_i(\sigma(y_1), \ldots, \sigma(y_i))$.*

The following proposition is straightforward.

Proposition 2.3 *A* QCSP *instance ϕ is true if and only if $\exists$ has a winning strategy in ϕ.*

It is well-known that the QCSP and CSP decision problems, in the general formulations given above, are **PSPACE**-complete and **NP**-complete, respectively. The broad research problem we focus on in this paper is to classify the complexity of the following parameterized version of the QCSP, for all possible parameterizations.

Definition 2.4 *Let $\Gamma \subseteq R_D$. The decision problems* CSP(Γ) *and* QCSP(Γ) *are restrictions of* CSP *and* QCSP*, respectively, to instances in which all predicates belong to Γ.*

We will now describe several combinatorial games that can be cast as QCSP(Γ) for a suitable set Γ.

Example 2.5 [Not-All-Equal 2-Colouring Game] An instance of this game is given by a linearly ordered set A and a collection C of (at most) three-element subsets of A. The players colour, in turn, elements of A with two colours, black and white, according to the ordering of A. Player $\exists$ wins if and only if, after all elements in A are coloured, each set in C has elements of both colours.

This game exactly corresponds to the problem QCSP($\{\varrho_{nae}\}$) where ϱ_{nae} is the ternary relation on $\{0, 1\}$ defined by $\varrho_{nae} = \{0, 1\}^3 \setminus \{(0, 0, 0), (1, 1, 1)\}$. □

Example 2.6 [One-In-Three 2-Colouring Game] An instance of this game is the same as in the preceding example. Player ∃ wins if and only if, after all elements in A are coloured, each set in C has exactly one black element.

This game exactly corresponds to the problem QCSP($\{\varrho_{1in3}\}$) where ϱ_{1in3} is the ternary relation on $\{0, 1\}$ defined by $\varrho_{1in3} = \{(0, 0, 1), (0, 1, 0), (1, 0, 0)\}$. □

Example 2.7 [Graph k-Colouring Game] In this game, an instance is a graph whose vertices are linearly ordered. In each move, a player colours one of the vertices in one of k colours. The order of moves is specified by the ordering on the vertices. Player ∃ wins if and only if, after all vertices are coloured, no adjacent vertices are of the same colour.

This game precisely corresponds to QCSP($\{\neq_k\}$) where $\neq_k$ is the disequality predicate on a set D such that $|D| = k$. To see this, consider the vertices of an input graph as variables, then the constraints are of the form $\neq_k (x, y)$ where (x, y) runs through the set of edges of the graph. □

Note that the game described in Example 2.7 is different from the graph colouring games of Bodlaender [2], where both players must satisfy all constraints by their moves, and the loser is the one who cannot make a move.

Example 2.8 [One-Or-Both Colour Matching Game] In this game an instance is a directed graph G whose nodes are linearly ordered, together with a set D (of colours). In addition, every arc (x, y) of G has a label which is a (suggested) pair a, b of colours from D for x and y, respectively. In each move, a player colours one of the nodes of G with some colour from D. The order of moves is specified by the ordering on the vertices. Player ∃ wins if and only if, after all vertices are coloured, the suggested pair of colours on every arc (x, y) matches at least one of the actual colours given to x, y.

This game precisely corresponds to QCSP(Γ_{cm}) where $\Gamma_{cm} \subseteq R_D$ consists of all binary relations of the form $\varrho_{a,b} = \{(u, v) \mid u = a \vee v = b\}$, for $a, b \in D$. □

Example 2.9 [Colour Implication Game] In this game an instance is a directed graph G whose nodes are linearly ordered, together with a set D (of colours) containing two distinguished colours, black and white. In addition, every arc (x, y) of G has a label which is a (suggested) pair a, b of non-distinguished colours from D for x and y, respectively. In each move, a player assigns a colour c_x to a vertex x of G. The order of moves is specified by the ordering on the vertices. Player ∃ wins if and only if, after all nodes are coloured, every arc $e = (x, y)$ satisfies the following condition: if c_x is black or matches its suggested colour, then c_y is also black or matches its suggested colour.

This game precisely corresponds to QCSP(Γ_{ci}) where $\Gamma_{ci} \subseteq R_D$ consists of all binary relations of the form $\varrho_{a,b} = \{(u, v) \mid u \in \{a, \text{black}\} \Rightarrow v \in \{b, \text{black}\}\}$, for $a, b \in D \setminus \{\text{black}, \text{white}\}$. □

Example 2.10 [Linear Equations Game] In this game an instance consists of a system of linear equations over a finite field K where the variables in the system are linearly ordered. The players alternately assign elements of K to the variables in the specified order. Player ∃ wins if and only if the obtained assignment is a solution to the system.

This game precisely corresponds to QCSP(Γ_{lin}) where $\Gamma_{lin} \subseteq R_K$ consists of all relations expressible by a linear equation over K. □

The results obtained in this paper will be sufficient to determine the complexity of deciding each of the six games described above (see Corollaries 3.9 and 8.1).

Finally, we observe that problems of the form CSP(Γ) with finite Γ can be expressed as homomorphism problems (see, for example, [26, 33]): in this formulation the question is to decide whether a given relational structure admits a homomorphism to a fixed relational structure. Hence all constraint satisfaction games can be viewed as particular examples of the following very general game.

Example 2.11 [HOMOMORPHISM CONSTRUCTION GAME] Fix an arbitrary relational structure $\mathcal{B} = (D; \varrho_1^{\mathcal{B}}, \ldots, \varrho_k^{\mathcal{B}})$ where $\varrho_i^{\mathcal{B}} \in R_D$ for all i. An instance of the game is another relational structure $\mathcal{A} = (V; \varrho_1^{\mathcal{A}}, \ldots, \varrho_k^{\mathcal{A}})$ such that the set V is linearly ordered and, for all $1 \leq i \leq k$, $\varrho_i^{\mathcal{A}} \in R_V$ and the relations $\varrho_i^{\mathcal{B}}$ and $\varrho_i^{\mathcal{A}}$ are of the same arity.

The players construct a mapping $h : V \to D$ by choosing, in turn and according to the order on V, images for elements of V. Player $\exists$ wins if and only if h is a homomorphism from $\mathcal{A}$ to $\mathcal{B}$, that is, for all $1 \leq i \leq k$, $h(\vec{x}) \in \varrho_i^{\mathcal{B}}$ whenever $\vec{x} \in \varrho_i^{\mathcal{A}}$.

This game precisely corresponds to QCSP(Γ) where $\Gamma = \{\varrho_1^{\mathcal{B}}, \ldots, \varrho_k^{\mathcal{B}}\}$. To see this, think of elements of V as variables, and, for every i and for every tuple $\vec{x} \in \varrho_i^{\mathcal{A}}$, introduce a constraint $\varrho_i^{\mathcal{B}}(\vec{x})$. As always, the order of moves corresponds to the order of quantifiers. □

3 Classifying Complexity

3.1 An algebraic approach

In earlier papers [12, 33, 35], an algebraic approach to studying the complexity of constraint satisfaction problems CSP(Γ) was developed (see also survey [39]). This approach is briefly reviewed in this subsection. We will use $O_D^{(n)}$ to denote the set of all n-ary operations on a set D (that is, the set of mappings $f\colon D^n \to D$), and O_D to denote the set $\bigcup_{n=1}^{\infty} O_D^{(n)}$. Any operation on D can be extended in a standard way to an operation on tuples over D, as follows. For any operation $f \in O_D^{(n)}$, and any collection of m-tuples $\vec{a}_1, \vec{a}_2, \ldots, \vec{a}_n \in D^m$, where $\vec{a}_i = (\vec{a}_i(1), \ldots, \vec{a}_i(m))$ (for $i = 1, \ldots, n$), define $f(\vec{a}_1, \ldots, \vec{a}_n)$ to be the m-tuple $(\, f(\vec{a}_1(1), \ldots, \vec{a}_n(1)), \ldots, f(\vec{a}_1(m), \ldots, \vec{a}_n(m))\,)$.

Definition 3.1 *For any relation $\varrho \in R_D^{(m)}$, and any operation $f \in O_D^{(n)}$, if $f(\vec{a}_1, \ldots, \vec{a}_n) \in \varrho$ for all choices of $\vec{a}_1, \ldots, \vec{a}_n \in \varrho$, then ϱ is said to be* invariant *under f, and f is called a* polymorphism *of ϱ.*

The set of all relations that are invariant under each operation from some set $C \subseteq O_D$ will be denoted $\mathsf{Inv}(C)$. The set of all operations that are polymorphisms of every relation from some set $\Gamma \subseteq R_D$ will be denoted $\mathsf{Pol}(\Gamma)$.

The following result provides a link between polymorphisms and the complexity of a CSP.

Theorem 3.2 ([33]) *Let Γ_1 and Γ_2 be sets of predicates over a finite set, such that Γ_1 is finite. If $\mathsf{Pol}(\Gamma_2) \subseteq \mathsf{Pol}(\Gamma_1)$ then* CSP(Γ_1) *is logarithmic-space reducible to* CSP(Γ_2).

This result shows that, when the set of values is finite, finite sets of predicates with the same polymorphisms give rise to constraint satisfaction problems which are mutually reducible to one another. In other words, the complexity of CSP(Γ) is determined by the polymorphisms of Γ. Note that Theorem 3.2 was originally stated in [33] with polynomial-time reducibility, but, by using the result of [45], it can easily be shown that the reduction is actually logarithmic-space.

The proof of Theorem 3.2 is made up of three crucial ingredients. The first is the fact that $\mathsf{Inv}(\cdot)$ and $\mathsf{Pol}(\cdot)$ form a Galois correspondence between R_D and O_D (see Proposition 1.1.14 of [43]). A basic introduction to this correspondence can be found in [42], and a comprehensive study in [43].

Proposition 3.3 *Let D be a finite set, $\Gamma, \Gamma' \subseteq R_D$, $C, C' \subseteq O_D$. Then*

$$\begin{array}{ll} \Gamma \subseteq \Gamma' \Longrightarrow \mathsf{Pol}(\Gamma) \supseteq \mathsf{Pol}(\Gamma') & C \subseteq C' \Longrightarrow \mathsf{Inv}(C) \supseteq \mathsf{Inv}(C') \\ \Gamma \subseteq \mathsf{Inv}(\mathsf{Pol}(\Gamma)) & C \subseteq \mathsf{Pol}(\mathsf{Inv}(C)) \\ \mathsf{Pol}(\Gamma) = \mathsf{Pol}(\mathsf{Inv}(\mathsf{Pol}(\Gamma))) & \mathsf{Inv}(C) = \mathsf{Inv}(\mathsf{Pol}(\mathsf{Inv}(C))) \end{array}$$

The second ingredient involves the set of predicates $\langle\Gamma\rangle$ defined below (see [33] for more information).

Definition 3.4 *For any set $\Gamma \subseteq R_D$ the set $\langle\Gamma\rangle$ consists of all predicates that can be expressed using*

1. *predicates from Γ, together with the binary equality predicate $=_D$ on D,*
2. *conjunction,*
3. *existential quantification.*

As the next proposition shows, the complexity of CSP(Γ) is, in effect, determined by $\langle\Gamma\rangle$; in particular, the problems CSP(Γ_1) and CSP(Γ_2) are of the same complexity if $\langle\Gamma_1\rangle = \langle\Gamma_2\rangle$.

Proposition 3.5 ([33, 12]) *Let Γ_1 and Γ_2 be sets of predicates over a finite set, such that Γ_1 is finite. If $\langle\Gamma_1\rangle \subseteq \langle\Gamma_2\rangle$, then* CSP($\Gamma_1$) *is logarithmic-space reducible to* CSP(Γ_2).

As with Theorem 3.2, this proposition was originally stated with polynomial-time reducibility, but it can be changed to logarithmic-space reducibility by using results of [45].

Finally, the third ingredient in the proof of Theorem 3.2 is the observation that the set $\langle\Gamma\rangle$ has an alternative characterization, which allows us to jump back to the polymorphisms of Γ.

Proposition 3.6 ([43]) *For any set of predicates Γ over a finite set, $\langle\Gamma\rangle = \mathsf{Inv}(\mathsf{Pol}(\Gamma))$.*

In Section 3.3 below, we will show that each of these three ingredients has an analog in the analysis of the QCSP.

3.2 Known classification results

A number of results on the complexity of constraint satisfaction problems have been obtained via the viewpoint of polymorphisms (see survey [39]). Indeed, we can re-state the classic dichotomy theorem of Schaefer [48] using the notion of polymorphism.

Theorem 3.7 ([48]) *For any $\Gamma \subseteq R_{\{0,1\}}$,* CSP($\Gamma$) *is in* **PTIME** *when* $\mathsf{Pol}(\Gamma)$ *contains one of the following:*

- *the constant 0 or constant 1 operations,*
- *the conjunction ($\wedge$) or disjunction ($\vee$) operations,*
- *the affine operation $x - y + z \pmod 2$,*
- *the majority operation $(x \vee y) \wedge (x \vee z) \wedge (y \vee z)$.*

In all other cases, CSP(Γ) *is* **NP***-complete.*

This reformulation of Schaefer's theorem can be demonstrated to follow from Theorem 3.2 and well-known algebraic results of Post [44]; see [33].

The complexity of QCSP(Γ) has also been completely classified in the Boolean case, giving an analog of Theorem 3.7 for quantified constraints.

Theorem 3.8 ([18, 19]) *For any* $\Gamma \subseteq R_{\{0,1\}}$*,* QCSP($\Gamma$) *is in* **PTIME** *when* $\mathsf{Pol}(\Gamma)$ *contains one of the following:*

- *the conjunction* ($\wedge$) *or disjunction* ($\vee$) *operations,*
- *the affine operation* $x - y + z \pmod 2$*,*
- *the majority operation* $(x \vee y) \wedge (x \vee z) \wedge (y \vee z)$*.*

In all other cases, QCSP(Γ) *is* **PSPACE***-complete.*

Using this theorem, it is not difficult to determine the complexity of the games described in Examples 2.5-2.10 in the special case of a two-valued domain.

Corollary 3.9 *In the special case when* $D = \{0, 1\}$*:*

(1) The games Not-All-Equal 2-Colouring *and* One-In-Three 2-Colouring *are* **PSPACE***-complete.*

(2) The games Graph 2-Colouring*,* One-Or-Both Colour Matching*,* Colour Implication *and* Linear Equations *can be decided in polynomial time.*

Proof: It is easy to verify that if Γ is $\{\varrho_{nae}\}$ or $\{\varrho_{1in3}\}$ then none of the four operations from Theorem 3.8 is a polymorphism of Γ. By Theorem 3.8, this proves part (1).

It is also straightforward to check that the majority operation is a polymorphism of every binary predicate on $\{0, 1\}$. Hence, by Theorem 3.8, the first three games listed in part (2) are polynomial-time decidable, since these games correspond to QCSPs with binary predicates. (Note that in the Colour Implication game we interpret black as 1 and white as 0). Finally, the affine operation is a polymorphism of any Boolean predicate given by a linear equation over GF(2), which shows that the fourth game listed in part (2) can also be decided in polynomial time. ∎

Theorem 3.8 was originally proved using combinatorial methods which do not easily generalize to larger sets of values. However, this theorem is most concisely stated using polymorphisms. In the next section we will show that, as with the complexity of CSP(Γ), for all finite sets of values the complexity of QCSP(Γ) depends only on the polymorphisms of Γ. In particular, we will show that a suitably modified version of Theorem 3.2 holds with QCSP(Γ) in place of CSP(Γ) (see Theorem 3.16). Thus, the algebraic approach of using polymorphisms to study complexity can also be applied to quantified constraints.

3.3 Surjective polymorphisms and the QCSP

As we noted in Section 3.1, the successful use of the algebraic approach to the CSP is possible due to three statements: Proposition 3.3, Proposition 3.5 and Proposition 3.6.

We now establish that in the case of the QCSP three similar properties hold for *surjective* polymorphisms. We thereby introduce a new Galois connection that involves surjective polymorphisms in place of arbitrary polymorphisms. We show that surjective polymorphisms play a similar role in the analysis of the QCSP to that played by arbitrary polymorphisms for the ordinary CSP (cf. Theorem 3.2). Let $\mathsf{s\text{-}Pol}(\Gamma)$ denote the set of all surjective operations contained in $\mathsf{Pol}(\Gamma)$. First, it is not hard to verify that the operators $\mathsf{Inv}(\cdot)$ and $\mathsf{s\text{-}Pol}(\cdot)$ form a Galois correspondence.

Proposition 3.10 *Let Γ, Γ' be sets of predicates on a finite set A and let C, C' be sets of surjective operations on A. Then*

$$\begin{array}{ll} \Gamma \subseteq \Gamma' \Longrightarrow \mathsf{s\text{-}Pol}(\Gamma) \supseteq \mathsf{s\text{-}Pol}(\Gamma') & C \subseteq C' \Longrightarrow \mathsf{Inv}(C) \supseteq \mathsf{Inv}(C') \\ \Gamma \subseteq \mathsf{Inv}(\mathsf{s\text{-}Pol}(\Gamma)) & C \subseteq \mathsf{s\text{-}Pol}(\mathsf{Inv}(C)) \\ \mathsf{s\text{-}Pol}(\Gamma) = \mathsf{s\text{-}Pol}(\mathsf{Inv}(\mathsf{s\text{-}Pol}(\Gamma))) & \mathsf{Inv}(C) = \mathsf{Inv}(\mathsf{s\text{-}Pol}(\mathsf{Inv}(C))) \end{array}$$

Next, we show that the complexity of QCSP(Γ) depends only on the set of predicates $[\Gamma]$, defined as follows.

Definition 3.11 *For any set $\Gamma \subseteq R_D$, the set $[\Gamma]$ consists of all predicates that can be expressed using*

1. *predicates from Γ, together with the binary equality predicate $=_D$ on D,*
2. *conjunction,*
3. *existential quantification,*
4. *universal quantification.*

We have the following parallel to Proposition 3.5.

Proposition 3.12 *Let Γ_1 and Γ_2 be sets of predicates over a finite set, such that Γ_1 is finite. If $[\Gamma_1] \subseteq [\Gamma_2]$, then* QCSP($\Gamma_1$) *is logarithmic-space reducible to* QCSP(Γ_2).

Proof: Let D be a finite set, and let $\Gamma_1, \Gamma_2 \subseteq R_D$. By Definition 3.11, for any n-ary relation ϱ in $[\Gamma_1]$, the predicate $\varrho(x_1, \ldots, x_n)$ is equivalent to a formula Φ_ϱ of the form $\mathcal{Q}_1 y_1 \ldots \mathcal{Q}_m y_m \mathcal{C}$, where the $\mathcal{Q}_i$, $1 \leq i \leq m$ are quantifiers, and $\mathcal{C}$ is a conjunction of atomic formulas involving only predicates from $\Gamma_2 \cup \{=_D\}$ and variables $x_1, \ldots, x_n$, $y_1, \ldots, y_m$.

Let sentence $\mathcal{P}_0$ be an instance of QCSP(Γ_1). Replace each predicate ϱ in $\mathcal{P}_0$ by the corresponding formula Φ_ϱ, to obtain an equivalent formula $\mathcal{P}_1$. Since Γ_1 is finite, this can be done in logarithmic space. Transform $\mathcal{P}_1$ into prenex normal form by moving all quantifiers (in order) to the front of the formula (renaming variables as needed to avoid name clashes). This transformation can also be carried out in logarithmic space. The resulting sentence, $\mathcal{P}_2$, is an instance of QCSP($\Gamma_2 \cup \{=_D\}$), and $\mathcal{P}_2$ is clearly equivalent to $\mathcal{P}_0$.

It now only remains to remove any occurrences of the equality relation from $\mathcal{P}_2$. We shall assume that $|D| \geq 2$ (the case $|D| = 1$ is trivial). Consider the graph $G = (V, E)$ whose vertices are the variables appearing in $\mathcal{P}_2$ and

$$E = \{(x, y) \in V^2 \mid (x = y) \text{ is a subformula in } \mathcal{P}_2\}.$$

For each connected component K, order the variables by the order in which they are quantified in $\mathcal{P}_2$. If K contains two variables, x and y, such that x is before y in this ordering and y is universally quantified, then $\mathcal{P}_2$ (and hence $\mathcal{P}_0$) is obviously false. Note that, by the result of [45], the existence of a path between two given vertices in an undirected graph can be decided in logarithmic space. In the remaining cases, all of the variables in K after the first must be existentially quantified, and because of the equality constraints they must all take the same value as the first variable. Hence, they can all be replaced with the first variable, removing the corresponding quantifiers, and removing the equality constraints. This can also be achieved in logarithmic space because we only need to check the existence of paths in G when transforming $\mathcal{P}_2$ as described above. After this procedure, we obtain a sentence $\mathcal{P}_3$ that is equivalent to $\mathcal{P}_0$ and is an instance of QCSP(Γ_2). Moreover, the whole transformation can be carried out in logarithmic space. ∎

The next example shows that Proposition 3.12 is stronger than Proposition 3.5 (reformulated for the QCSP in place of the CSP), as $[\Gamma]$ may be strictly larger than $\langle\Gamma\rangle$.

Example 3.13 Let ϱ be the relation $\{0,1\}^4 \setminus \{(0,0,0,1),(1,1,1,0)\}$, and let ϱ_{nae} be the relation $\{0,1\}^3 \setminus \{(0,0,0),(1,1,1)\}$ (as in Example 2.5). By "universally quantifying away" the last coordinate of ϱ, we obtain $\varrho_{nae} \in [\{\varrho\}]$. On the other hand, because the constant 0 and constant 1 operations are both polymorphisms of ϱ, every relation in $\langle\{\varrho\}\rangle$ contains both the "all-zeroes" tuple $(0,\ldots,0)$ and the "all-ones" tuple $(1,\ldots,1)$ by Theorem 3.6. It follows immediately that $\varrho_{nae} \notin \langle\{\varrho\}\rangle$. Setting $\Gamma_1 = \{\varrho_{nae}\}$ and $\Gamma_2 = \{\varrho\}$, we observe that $\Gamma_1 \not\subseteq \langle\Gamma_2\rangle$, and $\Gamma_1 \subseteq [\Gamma_2]$, giving examples of predicate sets Γ_1, Γ_2 such that the hypothesis of Proposition 3.12 holds, but the hypothesis of Proposition 3.5 does not. (Note that $\Gamma_1 \subseteq \langle\Gamma_2\rangle$ if and only if $\langle\Gamma_1\rangle \subseteq \langle\Gamma_2\rangle$; likewise, $\Gamma_1 \subseteq [\Gamma_2]$ if and only if $[\Gamma_1] \subseteq [\Gamma_2]$.) $\square$

In fact, we can use Proposition 3.12 to exhibit an example of a predicate giving rise to trivial CSPs, but also giving rise to intractable QCSPs.

Example 3.14 Let the relations ϱ and ϱ_{nae} be defined as in Example 3.13. Since $\varrho_{nae} \in [\{\varrho\}]$, Proposition 3.12 implies that QCSP($\{\varrho_{nae}\}$) reduces to QCSP($\{\varrho\}$), so by Corollary 3.9, QCSP($\{\varrho\}$) is **PSPACE**-complete. On the other hand, CSP($\{\varrho\}$) is trivial, as any instance is satisfiable by the "all-zeroes" or "all-ones" assignment. $\square$

Proposition 3.12 demonstrates the importance of the set $[\Gamma]$ with respect to the complexity of QCSP(Γ). By analogy to Theorem 3.6, $[\Gamma]$ can also be characterized in terms of polymorphisms.

Proposition 3.15 *For any set of predicates Γ over a finite set,* $[\Gamma] = \mathsf{Inv}(\mathsf{s\text{-}Pol}(\Gamma))$.

Proof: Let D be a finite set, and let $\Gamma \subseteq R_D$. The equality relation, $=_D$, is invariant under every operation on D, so $\Gamma \cup \{=_D\} \subseteq \mathsf{Inv}(\mathsf{s\text{-}Pol}(\Gamma))$. Let f be a surjective operation on D. It is straightforward to verify that applying conjunction or any quantification to predicates invariant under f gives another predicate which is also invariant under f. Hence, $[\Gamma] \subseteq \mathsf{Inv}(\mathsf{s\text{-}Pol}(\Gamma))$. Moreover, it follows that $\mathsf{s\text{-}Pol}(\Gamma) = \mathsf{s\text{-}Pol}([\Gamma])$.

To establish that $[\Gamma] \supseteq \mathsf{Inv}(\mathsf{s\text{-}Pol}(\Gamma))$, we will show that for any m-ary relation $\varrho \in \mathsf{Inv}(\mathsf{s\text{-}Pol}(\Gamma))$, the relation σ is a member of $[\Gamma]$, where σ is defined by

$$\sigma = \{(a_1, a_2, \ldots, a_m, d_1, d_2, \ldots, d_{|D|}) \mid (a_1, \ldots, a_m) \in \varrho, (d_1, \ldots, d_{|D|}) \in D^{|D|}\}.$$

From this it follows that $\varrho \in [\Gamma]$, by existentially quantifying over the last $|D|$ variables in σ.

To show that $\sigma \in [\Gamma]$, we first define $\sigma' = \bigcap\{\gamma \in [\Gamma] \mid \sigma \subseteq \gamma\}$. Since $[\Gamma]$ contains the total relation $D^{m+|D|}$, and is closed under conjunction, σ' is a member of $[\Gamma]$ and $\sigma \subseteq \sigma'$. In fact, σ' is the minimal relation of arity $m + |D|$ in $[\Gamma]$ with this property (when ordered by inclusion).

Now choose any tuple $\vec{c} = (b_1, \ldots, b_m, d_1, d_2, d_3, \ldots, d_{|D|}) \in \sigma'$. Note that σ' must also contain all tuples of the form $(b_1, \ldots, b_m, d'_1, d'_2, d'_3, \ldots, d'_{|D|})$, for each possible choice of $d'_1, d'_2, d'_3, \ldots, d'_{|D|}$, since otherwise we could obtain a smaller relation σ'' containing σ, by applying a sequence of universal quantifications, followed by a conjunction with the total relation $D^{m+|D|}$. Hence we may choose $\vec{c}$ so that the values of the d_i are all distinct, that is, $\{d_1, d_2, \ldots, d_{|D|}\} = D$.

By Definition 3.11, $[\Gamma]$ is closed under conjunction and existential quantification, and contains the equality relation, $=_D$. It is well-known (see Theorems 1.2.3 and 2.1.3 in [43]), that such sets satisfy the condition $[\Gamma] = \mathsf{Inv}(\mathsf{Pol}([\Gamma]))$. Furthermore, Proposition 1.1.19 of [43] implies that

$$\sigma' = \{f(\vec{a}_1, \ldots, \vec{a}_n) \mid n \geq 1,\ \vec{a}_1, \ldots, \vec{a}_n \in \sigma,\ f \in \mathsf{Pol}([\Gamma])\}.$$

Therefore, there exist $n \geq 1$, $\vec{a}_1, \ldots, \vec{a}_n \in \sigma$ and an n-ary function $f \in \mathsf{Pol}([\Gamma])$ such that $\vec{c} = f(\vec{a}_1, \ldots, \vec{a}_n)$.

By the choice of $\vec{c}$, the function f must be surjective. Therefore f is in $\mathsf{s\text{-}Pol}([\Gamma])$, and so $f \in \mathsf{s\text{-}Pol}(\Gamma)$, by the observation above. By the choice of ϱ, this implies that ϱ is invariant under f, and so $(b_1, \ldots, b_m) \in \varrho$. It follows that $\sigma' = \sigma$, so $\sigma \in [\Gamma]$, as required. ∎

We can now conclude that the complexity of QCSP(Γ) depends only on $\mathsf{s\text{-}Pol}(\Gamma)$, the surjective polymorphisms of Γ. The following theorem follows immediately from Propositions 3.10, 3.12 and 3.15.

Theorem 3.16 *Let Γ_1 and Γ_2 be sets of predicates over a finite set, such that Γ_1 is finite. If $\mathsf{s\text{-}Pol}(\Gamma_2) \subseteq \mathsf{s\text{-}Pol}(\Gamma_1)$, then QCSP($\Gamma_1$) is logarithmic-space reducible to QCSP(Γ_2).*

This theorem offers a dual perspective on the phenomenon displayed by Example 3.14, whereby a predicate set Γ can simultaneously give rise to a trivial CSP and give rise to an intractable QCSP. What is occurring is that the operations in $\mathsf{Pol}(\Gamma)$ that guarantee tractability of CSP(Γ) are non-surjective, and hence are not present in $\mathsf{s\text{-}Pol}(\Gamma)$.

4 Tractability

Comparing the statements of Theorems 3.7 and 3.8, we observe that, in two-valued domains, surjective polymorphisms which ensure the tractability of the CSP also ensure the tractability of the QCSP. However, it certainly cannot be taken for granted that a similar statement holds for non-Boolean domains. In this section, we show that it does hold for two broad classes of surjective polymorphisms.

4.1 Mal'tsev operations

An operation $m(x, y, z)$ on D is said to be *Mal'tsev* if it satisfies the identities $m(x, y, y) = m(y, y, x) = x$ for all x, y. For example, for an Abelian group G, the operation $f(x, y, z) = x - y + z$, called the *affine operation* of G, is a Mal'tsev operation. Relations invariant under the affine operation of a finite Abelian group play a significant role in the study of the complexity of the standard constraint satisfaction problem [26, 33, 35].

Let $\Gamma = \mathsf{Inv}(\{m\})$, where m is some fixed Mal'tsev operation. A polynomial-time algorithm for solving CSP(Γ) was given in [10]. Moreover, this algorithm also finds a satisfying assignment for any satisfiable instance of CSP(Γ). We will show now that QCSP(Γ) can also be solved in polynomial time by making repeated use of this algorithm.

Lemma 4.1 *Let m be a Mal'tsev operation on a finite set D, let $\mathcal{P} = \mathcal{Q}_1 x_1 \ldots \mathcal{Q}_n x_n\ \psi(x_1, \ldots, x_n)$ be an instance of QCSP($\mathsf{Inv}(\{m\})$), and let j be the maximal index such that $\mathcal{Q}_j$ is the universal quantifier.*

(1) If $\psi'(x_1, \ldots, x_{j-1}) = \forall x_j \exists x_{j+1} \ldots \exists x_n\ \psi(x_1, \ldots, x_n)$ is satisfiable then, for any model $(c_1, \ldots, c_n)$ of ψ, the tuple $(c_1, \ldots, c_{j-1})$ is a model of ψ'.

(2) $\mathcal{P}$ is true if and only if $\mathcal{P}' = \mathcal{P}_1 \wedge \mathcal{P}_2$ is true, where

$$\begin{aligned}\mathcal{P}_1 &= \mathcal{Q}_1 x_1 \ldots \mathcal{Q}_{j-1} x_{j-1} \exists x_j \exists x_{j+1} \ldots \exists x_n\ \psi(x_1, \ldots, x_n),\\ \mathcal{P}_2 &= \exists x_1 \ldots \exists x_{j-1} \forall x_j \exists x_{j+1} \ldots \exists x_n\ \psi(x_1, \ldots, x_n).\end{aligned}$$

Proof: (1) Let $(a_1, \ldots, a_{j-1})$ be a model for ψ', and for each $b \in D$, let $(a_j^b, \ldots, a_n^b)$ be an extension such that $\vec{a}^b = (a_1, \ldots, a_{j-1}, a_j^b, \ldots, a_n^b)$ is a model for ψ and $a_j^b = b$.

Take an arbitrary model $\vec{c} = (c_1, \ldots, c_{j-1}, c_j, \ldots, c_n)$ of ψ. We need to show that $(c_1, \ldots, c_{j-1})$ is a model of ψ'. Fix an arbitrary $b \in D$ and let $\vec{d} = (d_1, \ldots, d_n)$ be equal to $m(\vec{a}^b, \vec{a}^{c_j}, \vec{c})$. Proposition 3.15 implies that the predicate defined by ψ is invariant under m, so $\vec{d}$ is a model of ψ, too. Moreover, we have $d_i = m(a_i, a_i, c_i) = c_i$ for $i \in \{1, \ldots, j-1\}$ and $d_j = m(a_j^b, a_j^{c_j}, c_j) = m(b, c_j, c_j) = b$. Thus, $(c_1, \ldots, c_{j-1})$ is a model of ψ'.

(2) Obviously, if $\mathcal{P}$ is true then $\mathcal{P}'$ is also true. The inverse implication easily follows from part (1). Indeed, since $\mathcal{P}_2$ is true, we can apply (1); then, (1) implies that every tuple $(c_1, \ldots, c_{j-1})$ that can be extended to a model of ψ can be extended so with c_j being any given element. Thus, since $\mathcal{P}_1$ is true, so is $\mathcal{P}$. ∎

Theorem 4.2 *Let m be an arbitrary Mal'tsev operation on a finite set D. The problem class* QCSP($\mathsf{Inv}(\{m\})$) *is in* **PTIME**.

Proof: Let $\mathcal{P} = \mathcal{Q}_1 x_1 \ldots \mathcal{Q}_n x_n \ \psi(x_1, \ldots, x_n)$ be an instance of the problem class QCSP($\mathsf{Inv}(\{m\})$). By repeatedly applying Lemma 4.1(2), one can show that $\mathcal{P}$ can be decomposed into a conjunction of instances which have the same quantifier-free part and each contain at most one universal quantifier. Moreover, if we can find a model of ψ then Lemma 4.1(1) implies that initial segments of this model can be used in deciding whether each of the instances is true. It remains to notice that, as is easy to check, fixing a value for any variable in a predicate from $\mathsf{Inv}(\{m\})$ gives another predicate invariant under m, which implies that $\exists x_{l+1} \ldots \exists x_n \psi(c_1, \ldots, c_{l-1}, b, x_{l+1}, \ldots, x_n)$ is also an instance of CSP($\mathsf{Inv}(\{m\})$). Now it follows that the algorithm shown in Figure 1 is correct.

Input $\mathcal{P} = \mathcal{Q}_1 x_1 \ldots \mathcal{Q}_n x_n \ \psi(x_1, \ldots, x_n)$ where $\psi = \varrho_1 \wedge \ldots \wedge \varrho_q$, and $\varrho_1, \ldots, \varrho_q \in \Gamma$.
Output 'YES' if $\mathcal{P}$ is true, 'NO' otherwise.

Step 1 **solve** the instance $\exists x_1 \ldots \exists x_n \ \psi$
Step 2 **if** ψ has a model **then find** one, $(c_1, \ldots, c_n)$
  else output('NO') and **stop**
Step 3 **for** $l = n, \ldots, 1$ **do**
Step 3.1 **if** $\mathcal{Q}_l$ is the universal quantifier **then**
Step 3.2 **for each** $b \in D$ **do**
Step 3.2.1 **solve** the instance $\exists x_{l+1} \ldots \exists x_n \psi(c_1, \ldots, c_{l-1}, b, x_{l+1}, \ldots, x_n)$
Step 3.2.2 **If** this instance has no solution **then output**('NO') and **stop**
  enddo
  enddo
Step 4 **output**('YES')

Figure 1: Algorithm for deciding QCSP(Γ) when Γ has a Mal'tsev polymorphism

This algorithm uses $k|D| + 1$ applications of an algorithm for solving CSP($\mathsf{Inv}(\{m\})$), where k is the number of universal quantifiers in an instance, and one application of an algorithm finding a model. Now we can use the polynomial-time algorithm for CSP($\mathsf{Inv}(\{m\})$) developed in [10]. This completes the proof of Theorem 4.2. ∎

Note that if the operation m has a special form then the method described above can be used to derive more specialised, and more efficient, algorithms. For example, let G be a finite Abelian group, with affine operation f, and unit element 0, and let Γ be a finite set of relations over G which are invariant under f. Note that, by straightforward algebraic manipulation, it can be shown that any (n-ary) relation invariant under f is a coset of a subgroup of the group G^n.

In the simplest case, when the order of G is prime, G can be considered as a prime field, and hence G^n can be considered as a vector space over G. In this case, each coset of a subgroup of G^n is a linear variety, and it is well-known that such varieties can be defined by systems of linear equations, whose coefficients are elements of the field G. Therefore, in this case, QCSP(Γ) can be considered as the problem of solving quantified linear systems over G, which can be done by applying standard techniques from linear algebra, or by using them in the above algorithm.

In the case when G is an arbitrary Abelian group, QCSP(Γ) requires a similar but slightly more involved algorithm, see [5].

4.2 Near-unanimity operations

Our second example of surjective polymorphisms which give rise to tractable quantified constraint satisfaction problems concerns operations known as *near-unanimity operations*. An operation $f : D^k \to D$ is said to be *near-unanimity* if $k \geq 3$ and f returns the value a whenever at least $k-1$ of its arguments are equal to a; that is, for all $a, b \in D$, it holds that $a = f(b, a, a, \ldots, a, a, a) = f(a, b, a, \ldots, a, a, a) = \ldots = f(a, a, a, \ldots, a, b, a) = f(a, a, a, \ldots, a, a, b)$.

Before giving the tractability result for the QCSP associated with such polymorphisms, we introduce some notions of *consistency*.

Definition 4.3 ([34]) *Let ψ be an instance of the CSP with variable set $V = \{x_1, \ldots, x_n\}$. For a subset V' of V, a mapping $g' : V' \to D$ is a* partial solution *to ψ if, for every atomic formula $\varrho(\vec{v})$ from ψ, there is an extension $g : V \to D$ of g' satisfying $\varrho(\vec{v})$. The formula ψ is* j-consistent *if, for every subset V' of V with $|V'| = j-1$ and for every variable $v \in V \setminus V'$, any partial solution $g' : V' \to D$ of ψ can be extended to a partial solution $g' : V' \cup \{v\} \to D$ of ψ. The instance ψ is* strongly k-consistent *if it is j-consistent for $j = 2, \ldots, k$. The set ψ is* globally consistent *if it is strongly $|V|$-consistent.*

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

Theorem 4.4 ([34]) *Let f be an arbitrary near-unanimity operation of arity r on a finite set D. Any instance of* CSP($\mathsf{Inv}(\{f\})$) *which is strongly r-consistent is globally consistent.*

Theorem 4.4 implies that invariance under a near-unanimity operation implies CSP tractability, as any CSP instance ψ can be transformed into one that is strongly r-consistent in polynomial time, as follows. Take any pair $\varrho_s(\vec{v}_s)$ and $\varrho_t(\vec{v}_t)$ of atomic formulas in ψ, and any set of variables U, $|U| \leq r-1$, shared by these two atomic formulas. Replace $\varrho_s(\vec{v}_s)$ by the atomic formula $\varrho'_s(\vec{v}_s)$ which is equivalent to $\exists y_1, \ldots, y_p(\varrho_s(\vec{v}_s) \wedge \varrho_t(\vec{v}_t))$ where $y_1, \ldots, y_p$ are the variables that appear in $\vec{v}_t$ but do not belong to U. Similarly, replace $\varrho_t(\vec{v}_t)$ by the atomic formula $\varrho'_t(\vec{v}_t)$ which is equivalent to $\exists z_1, \ldots, z_q(\varrho_s(\vec{v}_s) \wedge \varrho_t(\vec{v}_t))$ where $z_1, \ldots, z_p$ are the variables that appear in $\vec{v}_s$ but do not belong to U. Do this for every pair of atomic formulas, and for every set of variables U, $|U| \leq r-1$, shared by the atomic formulas, until the process stabilizes. Since r is a constant, the process will stabilize after polynomially many steps, and it can be straightforwardly checked that the obtained CSP instance is strongly r-consistent and has the same set of satisfying assignments as ψ.

We prove that invariance under a near-unanimity operation implies QCSP tractability as well.

Theorem 4.5 *Let f be an arbitrary near-unanimity operation on a finite set D. The problem class* $\mathrm{QCSP}(\mathsf{Inv}(\{f\}))$ *is in* **PTIME**.

Proof: Let f be a near-unanimity operation of arity r, and let $\mathcal{P} = \mathcal{Q}_1 x_1 \ldots \mathcal{Q}_n x_n \psi$ be an instance of the $\mathrm{QCSP}(\mathsf{Inv}(\{f\}))$ problem. We show that a new instance $\mathcal{P}'$ of the $\mathrm{QCSP}(\mathsf{Inv}(\{f\}))$ problem can be computed in polynomial time, where $\mathcal{P}'$ has one fewer quantifier than $\mathcal{P}$ and does not have a greater number of predicates than $\mathcal{P}$. Moreover, the instance $\mathcal{P}'$ will have the property that it is valid if and only if $\mathcal{P}$ is valid. This suffices to establish that $\mathrm{QCSP}(\Gamma)$ is in **PTIME**, since the procedure can be iteratively applied to decide the validity of an instance of $\mathrm{QCSP}(\Gamma)$.

The new formula $\mathcal{P}'$ is obtained from $\mathcal{P}$ by elimination of the innermost quantifier and associated quantified variable, $\mathcal{Q}_n x_n$. We split into two cases depending on the type of $\mathcal{Q}_n$.

Case $\mathcal{Q}_n = \exists$: Obtain ψ_0 from ψ by establishing strong r-consistency; this can be done in polynomial time. As the procedure for establishing strong r-consistency involves only replacing predicates with those obtained from them by using conjunction and existential quantification, all predicates in ψ_0 are invariant under f. We next create a formula ψ' by including in it every constraint from ψ_0, but "projecting out" the variable x_n from any constraints where it is present. More precisely, we create ψ' as follows: for every constraint $\varrho_0(\vec{v}_0)$ in ψ_0, if $\vec{v}_0$ does not contain x_n, then include the constraint $\varrho_0(\vec{v}_0)$ in ψ'; otherwise, include in ψ' the atomic formula equivalent to $\exists x_n \varrho_0(\vec{v}_0)$.

By Theorem 4.4, any satisfying assignment for ψ' can be extended to a satisfying assignment for ψ_0. Moreover, any satisfying assignment for ψ_0 is straightforwardly verified to be a satisfying assignment for ψ', from the definition of ψ'. We therefore have that $\psi' = \exists x_n \psi_0$, and may define $\mathcal{P}'$ to be $\mathcal{Q}_1 x_1 \ldots \mathcal{Q}_{n-1} x_{n-1} \psi'$.

Case $\mathcal{Q}_n = \forall$: Create a formula ψ' from the formula ψ as follows: replace each constraint $\varrho(\vec{v})$ in ψ by the atomic formula equivalent to $\forall x_n \varrho(\vec{v})$. Since f is surjective, every predicate in the new formula is invariant under f. We may therefore define $\mathcal{P}' = \mathcal{Q}_1 x_1 \ldots \mathcal{Q}_{n-1} x_{n-1} \psi'$. ∎

Note that Theorem 4.5 can be strengthened for a special ternary near-unanimity operation known as the *dual discriminator*, which is the operation d such that $d(x, y, z) = y$ if $y = z$ and $d(x, y, z) = x$ otherwise. It was shown in [5] that $\mathrm{QCSP}(\mathsf{Inv}(\{d\}))$ belongs to complexity class **NL**.

5 Intractability

In this section we will use Theorem 3.16 to give a sufficient condition, in terms of surjective polymorphisms, for **PSPACE**-completeness of $\mathrm{QCSP}(\Gamma)$. We first establish that a particular QCSP problem is **PSPACE**-complete. This problem corresponds to a generalized form of the standard GRAPH-$|D|$-COLORABILITY problem [29, 41] (see Example 2.7).

Proposition 5.1 $\mathrm{QCSP}(\{\neq_D\})$ *is* **PSPACE**-*complete when* $|D| \geq 3$.

Proof: We prove this by reduction from $\mathrm{QCSP}(\{\varrho_{nae}\})$, where ϱ_{nae} is the ternary not-all-equal predicate on a 2-element set, as defined in Example 2.5. Let $\mathcal{P}$ be an instance of $\mathrm{QCSP}(\{\varrho_{nae}\})$, with variables $v_1, v_2, \ldots, v_n$. We construct a corresponding instance $\mathcal{P}'$ of $\mathrm{QCSP}(\{\neq_D\})$ as follows.

First construct a graph, $G_{\mathcal{P}}$, as shown in Fig. 2, with 3 nodes for each variable v_i of $\mathcal{P}$ (labelled x_i, y_i and z_i), 3 nodes for each triple of variables constrained by ϱ_{nae} in $\mathcal{P}$, and one additional node (labelled w). Connect these nodes as indicated in Fig. 2, so that each z_i is connected to y_i, each y_i is connected to x_i and w, and each x_i is connected to w. For each triple of nodes representing a constraint on $v_{i_1}, v_{i_2}, v_{i_3}$, connect these nodes to form a triangle and also add edges from these nodes to the corresponding nodes $x_{i_1}, x_{i_2}, x_{i_3}$.

The standard 3-colouring problem for the graph $G_{\mathcal{P}}$ can be represented as the satisfiability problem for the formula built as follows: introduce a variable for each node of the graph, and form a conjunction which contains a binary disequality constraint $\neq_D(u, v)$ for each edge (u, v) of the graph. (If D contains more than 3 values, then we add a clique C containing $|D| - 3$ nodes to the graph $G_{\mathcal{P}}$ and connect each node, except the nodes z_i, in the original graph to each node of this clique. This ensures that each node, except the nodes z_i, in the original graph $G_{\mathcal{P}}$ must be coloured with one of the 3 colours not used to colour C.)

Now add quantifiers to this conjunction of constraints as follows. First existentially quantify the variable w (and all the variables corresponding to nodes of the clique C, if present). Note that once values have been assigned to these nodes there are just two remaining possible values for each node x_i and y_i.

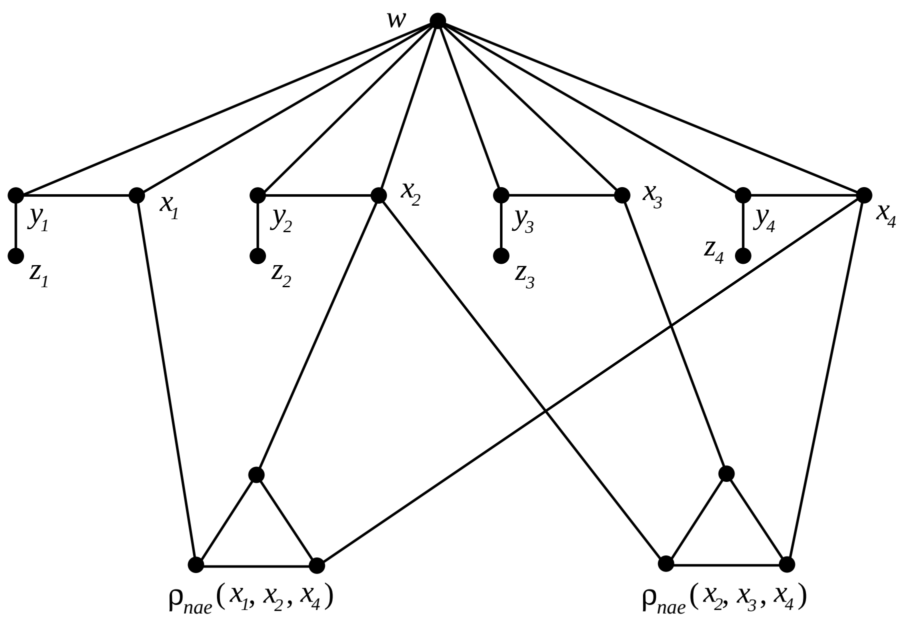

Figure 2: The construction used in the proof of Proposition 5.1 (adapted from Figure 9.8 in [41])

Next, for each consecutive quantifier of $\mathcal{P}$ (in order) introduce 3 consecutive quantifiers as follows:

- For each existential quantifier in $\mathcal{P}$, $\exists v_i$, introduce $\exists z_i \exists y_i \exists x_i$;
- For each universal quantifier in $\mathcal{P}$, $\forall v_i$, introduce $\forall z_i \exists y_i \exists x_i$.

Finally, add existential quantifiers for all remaining variables (corresponding to the constraints of $\mathcal{P}$). This completes the definition of $\mathcal{P}'$.

It is straightforward to check that there is an assignment of Boolean values to the variables $v_1, v_2, \ldots, v_n$ satisfying all of the constraints of $\mathcal{P}$ if and only if there is an assignment of values from D to the variables of $\mathcal{P}'$ satisfying all the constraints of $\mathcal{P}'$. This is because to satisfy the constraints of $\mathcal{P}'$, the 3 nodes in each triangle in $G_{\mathcal{P}}$ corresponding to a constraint of $\mathcal{P}$ must all be assigned distinct values, which is possible if and only if the corresponding nodes x_{i_1}, x_{i_2} and

x_{i_3} connected to them do not all take the same value (which mimics satisfying assignments for the constraint $\varrho_{nae}(v_{i_1}, v_{i_2}, v_{i_3})$). Furthermore, the construction of the quantifiers in $\mathcal{P}'$ ensures that the sentence $\mathcal{P}'$ is true if and only if $\mathcal{P}$ is true. To see this, note that for any variable v_i of $\mathcal{P}$ which is universally quantified, the universal quantification on the corresponding z_i in $\mathcal{P}'$ forces y_i (and, hence, x_i) to take both remaining available values, which mimics the universal quantification on v_i.

Hence, we have established a reduction from QCSP($\{\varrho_{nae}\}$) to QCSP($\{\neq_D\}$), and it is clear that this reduction can be carried out in logarithmic space. Since QCSP($\{\varrho_{nae}\}$) is **PSPACE**-complete, by Corollary 3.9, the result follows. ∎

Theorem 5.2 *For any finite set D with $|D| \geq 3$, and any $\Gamma \subseteq R_D$, if every $f \in \mathsf{s\text{-}Pol}(\Gamma)$ is of the form $f(x_1, \ldots, x_n) = g(x_i)$ for some $1 \leq i \leq n$ and some permutation g on D, then* QCSP(Γ) *is* **PSPACE***-complete.*

Proof: By Lemma 1.3.1 (b) of [43], $\mathsf{Pol}(\{\neq_D\})$, for $|D| \geq 3$, consists of all operations of the form described in the Theorem. Hence $\mathsf{Pol}(\{\neq_D\}) = \mathsf{s\text{-}Pol}(\{\neq_D\})$, and we can apply Theorem 3.16 and Proposition 5.1. ∎

We now give an example of a relation which has all possible non-surjective polymorphisms, but whose surjective polymorphisms are precisely the operations described in Theorem 5.2.

Example 5.3 Let τ_s be the s-ary "not-all-distinct" predicate holding on a tuple $(a_1, \ldots, a_s)$ if and only if $|\{a_1, \ldots, a_s\}| < s$. Note that $\tau_s \supseteq \{(a, \ldots, a) \mid a \in D\}$, so every instance of CSP($\{\tau_s\}$) is trivially satisfiable by assigning the same value to all variables.

However, by Lemma 2.2.4 of [43], the set $\mathsf{Pol}(\{\tau_{|D|}\})$ consists of all possible non-surjective operations on D, together with all operations of the form given in Theorem 5.2. Hence, $\{\tau_{|D|}\}$ satisfies the conditions of Theorem 5.2, and QCSP($\{\tau_{|D|}\}$) is **PSPACE**-complete (when $|D| \geq 3$).

Interestingly, the predicate $\tau_{|D|}$ has the property that, for every predicate $\varrho \in \langle \tau_{|D|} \rangle \setminus \langle =_D \rangle$, we have $\langle \varrho \rangle = \langle \tau_{|D|} \rangle$ (Lemma 2.2.4 of [43]). □

6 Semilattice Operations

A *semilattice operation* $*$ on a set D is a binary operation that satisfies the following conditions for all $a, b, c \in D$:

(1) $*(a, a) = a$ (*idempotency*);

(2) $*(a, b) = *(b, a)$ (*commutativity*);

(3) $*(*(a, b), c) = *(a, *(b, c))$ (*associativity*).

Normally we shall use infix notation for semilattice operations and write $a * b$ rather than $*(a, b)$. As is easily seen, the propositional conjunction and disjunction operations are semilattice operations on the set $\{0, 1\}$.

It is well-known that every semilattice operation $*$ induces a partial order $\leq_*$, where $a \leq_* b$ if and only if $a * b = b$. For $a, b \in D$, the element $a * b$ is the *least upper bound* of a, b with respect to this order, i.e. $a, b \leq_* a * b$ and, for any $d \in D$ such that $a, b \leq_* d$, we have $a * b \leq_* d$.

Every semilattice operation $*$ has a *zero element* 0 with the property that $a \leq_* 0$, or equivalently $a * 0 = 0 * a = 0$, for all $a \in A$. If a semilattice operation also has a *unit element* — that is an element 1 such that $1 \leq_* a$, or equivalently $1 * a = a * 1 = a$, for all $a \in A$ — then we say that it is a *semilattice operation with unit* or a *monoid operation*; otherwise, we say that it is a *semilattice operation without unit.* Interestingly, if $*$ is a monoid operation then, for any $a, b \in D$, there is a unique *greatest lower bound* c of a, b with respect to this order, i.e. $c \leq_* a, b$ and, for any $d \in D$ such that $d \leq_* a, b$, we have $d \leq_* c$ (in other words, the order $\leq_*$ is a *lattice* order). The greatest lower bound of a, b will be denoted by $a \circ b$. Operation $*$ can be extended to an operation on tuples of elements from D in the usual way (by applying the operation componentwise).

All forms of semilattice operations were shown to guarantee CSP tractability in [35]. For the QCSP the situation is rather different. The following theorem completely classifies the complexity of the QCSP over a set of predicates invariant under a semilattice operation.

Theorem 6.1 *Let $*$ be a semilattice operation on a finite set D. If $*$ is an operation with unit then, for any finite $\Gamma \subseteq \mathsf{Inv}(*)$, the problem* QCSP($\Gamma$) *is in* **PTIME**. *Otherwise, there exists a finite $\Gamma \subseteq \mathsf{Inv}(*)$ such that* QCSP(Γ) *is* **PSPACE**-*complete.*

The proof of Theorem 6.1 is given in Sections 6.1 and 6.2 below.

6.1 Semilattice operations with unit

In this subsection, we demonstrate that finite sets of predicate which are invariant under a semilattice operation with unit give rise to tractable subproblems of the QCSP.

Our first step is to demonstrate that constraints which are invariant under a semilattice operation are decomposable into what we call Horn-like clauses. We introduce the following definitions and notation. A *downward literal* is an expression of the form $v \leq_* d$, where v is a variable and $d \in D$; and, an *upward literal* is an expression of the form $v \not\leq_* d$, where v is a variable and $d \in D$. We will call literals of the form $v \leq_* 0$ or $v \not\leq_* 0$ *trivial.* A literal occurring in a QCSP instance is an $\exists$-literal ($\forall$-literal) if its variable is an existentially (universally) quantified variable. A *Horn-like clause* is a set with downward and upward literals as elements which contains at most one downward literal. A Horn-like clause is interpreted as the disjunction of the literals that it contains; that is, it is considered to be true if at least one of its literals is true.

Lemma 6.2 *A predicate ϱ is invariant under a semilattice operation if and only if ϱ can be represented as a conjunction of Horn-like clauses.*

Proof: For any semilattice operation $*$, the element $v_1 * v_2$ is the least upper bound of v_1, v_2 with respect to the order $\leq_*$. Hence, $v_1 * v_2 \leq_* a$ if and only if $v_1 \leq_* a$ and $v_2 \leq_* a$. Using this result it is straightforward to verify that any Horn-like clause is invariant under the corresponding semilattice operation.

For the converse, let ϱ be invariant under a semilattice operation $*$. It suffices to show that for each $\vec{a} \notin \varrho$, there exists a Horn-like clause $H_{\vec{a}}$ such that any tuple from ϱ satisfies $H_{\vec{a}}$, but $\vec{a}$ does not satisfy $H_{\vec{a}}$. Fix $\vec{a} \notin \varrho$, and define $\sigma_{\vec{a}} = \{\vec{b} : \vec{b} \in \varrho, \vec{b} \leq_* \vec{a}\}$ where $\leq_*$ is extended to a partial ordering on tuples by defining $\vec{s} \leq_* \vec{t}$ if and only if at all coordinates i, $\vec{s}(i) \leq_* \vec{t}(i)$.

If $\sigma_{\vec{a}}$ is empty, then it can be verified that the Horn-like clause $H_{\vec{a}} = \bigcup_{i=1}^{r} \{\vec{v}(i) \not\leq_* \vec{a}(i)\}$ has the desired properties, where r denotes the length of the tuples $\vec{v}$ and $\vec{a}$. Otherwise, define $\vec{m} = \vec{a}_1 * \vec{a}_2 * \ldots * \vec{a}_n$, where $\sigma_{\vec{a}} = \{\vec{a}_1, \vec{a}_2, \ldots, \vec{a}_n\}$. It is straightforward to verify that $\sigma_{\vec{a}}$ is invariant under $*$, so $\vec{m} \in \sigma_{\vec{a}}$. Since $\sigma_{\vec{a}} \subseteq \varrho$, $\vec{m} \neq \vec{a}$, and $\vec{m}$ and $\vec{a}$ differ at some coordinate, say coordinate j (that is, $\vec{m}(j) \neq \vec{a}(j)$, which implies that $\vec{m}(j) <_* \vec{a}(j)$). Hence, it can be verified that the Horn-like clause $H_{\vec{a}} = (\bigcup_{i=1}^{r} \{\vec{v}(i) \not\leq_* \vec{a}(i)\}) \cup \{\vec{v}(j) \leq_* \vec{m}(j)\}$ has the desired properties. ∎

Lemma 6.2 generalizes Horn's classic theorem that a constraint with relation invariant under logical AND ($\wedge$) over the set $\{0, 1\}$ is logically equivalent to a conjunction of Horn clauses [32]. To see this, we let $* = \wedge$; then we have $1 \leq_* 0$ and the only non-trivial downward literal is one of the form $v \leq_* 1$, which is equivalent to the positive literal v. The only non-trivial upward literal is one of the form $v \not\leq_* 1$, which is equivalent to the negative literal $\overline{v}$.

Our next step is to define a proof system, called *QCSP-literal-resolution*, and show that it is sound and complete for quantified formulas consisting of Horn-like clauses, with respect to a semilattice operation $*$ with unit. We define a Horn-like clause H appearing in a QCSP $\mathcal{P}$ to be an *existential unit clause* if it contains only one $\exists$-literal, the single $\exists$-literal is downward, and for every $\forall$-variable y in H, y comes before the variable of the $\exists$-literal in the quantification order of the formula $\mathcal{P}$.

Definition 6.3 *Let $\mathcal{P}$ be a QCSP instance with quantifier-free part ψ where every predicate in ψ is a Horn-like clause (with respect to a semilattice operation $*$ with unit). We say that a Horn-like clause H is derivable by* QCSP-literal-resolution *from the formula ψ, denoted $\psi \vdash_l H$, if it can be obtained by applying the following rules.*

0. For every predicate H in ψ, $\psi \vdash_l H$.

1. If $\psi \vdash_l H_1$, $\psi \vdash_l H_2$, and there exist elements $a, b \in D$ such that $(x \leq_* a) \in H_1$ and $(x \leq_* b) \in H_2$, for some existentially quantified variable x, then

$$\psi \vdash_l (H_1 \setminus \{x \leq_* a\}) \cup (H_2 \setminus \{x \leq_* b\}) \cup \{x \leq_* a \circ b\}.$$

2. If $\psi \vdash_l H$ and $(x \leq_* a) \in H$ for some $a \in D$ and some existentially quantified variable x, then for all $b \in D$ such that $a \leq_* b$

$$\psi \vdash_l (H \setminus \{x \leq_* a\}) \cup \{x \leq_* b\}.$$

3. If $\psi \vdash_l U$ and $\psi \vdash_l H$, where U is an existential unit clause with downward literal $(x \leq_* a)$, and $(x \not\leq_* a) \in H$, then

$$\psi \vdash_l (U \setminus \{x \leq_* a\}) \cup (H \setminus \{x \not\leq_* a\}).$$

4. If $\psi \vdash_l H$, y is a universally quantified variable which is the last variable in the quantification order of ψ occurring in H, and there exists a value $a \in D$ such that the assignment $y = a$ does not satisfy H_y (the clause containing all y-literals in H), then

$$\psi \vdash_l H \setminus H_y.$$

Lemma 6.4 *Let $\mathcal{P}$ be a QCSP instance with quantifier-free part ψ such that every predicate in ψ is a Horn-like clause (with respect to a semilattice operation $*$ with unit) without trivial literals. The sentence $\mathcal{P}$ is false if and only if $\psi \vdash_l \emptyset$.*

Proof: Each of the rules listed in Definition 6.3 preserves satisfiability, so the "if" direction is straightforward, and we will focus on the "only if" direction. Assume without loss of generality that $\mathcal{P}$ has the form

$$\forall y_1 \exists x_1 \ldots \forall y_n \exists x_n \psi(y_1, x_1, \ldots, y_n, x_n),$$

and suppose that it is not the case that $\psi \vdash_l \emptyset$. Define (for $k = 1, \ldots, n$)

$$\tau_k(a_1, \ldots, a_k) = *\{\ a \mid \text{the assignment } y_1 = a_1, \ldots, y_k = a_k, x_k = a \text{ satisfies all existential unit clauses } C \text{ containing } x_k \text{ such that } \psi \vdash_l C\}.$$

Note that $\tau_k(a_1,\dots,a_k)=0$ if there is no derivable existential unit clause containing x_k. We claim that the mappings τ_k form a winning strategy. Let $f:\{y_1,\dots,y_n\}\to D$ be an assignment to the $\forall$-variables of ψ; we wish to show that the assignment

$$\tau_f(z)=\begin{cases} f(y_k) & \text{if } z=y_k,\\ \tau_k(f(y_1),\dots,f(y_k)) & \text{if } z=x_k\end{cases}$$

satisfies all clauses of ψ. By rule 0, it suffices to show that τ_f satisfies all clauses H such that $\psi\vdash_l H$. Recall that a Horn-like clause potentially contains four types of literals: upward $\exists$-literals, upward $\forall$-literals, downward $\exists$-literals and downward $\forall$-literals, but has at most one downward literal of either kind. We shall prove that τ_f satisfies all clauses H such that $\psi\vdash_l H$ by induction on the number, u, of upward $\exists$-literals contained in H.

We split into three cases.

Case 1: $u=0$ and H does not contain any downward $\exists$-literals either. In this case all literals of H are $\forall$-literals, so if τ_f does not satisfy H, then the empty set can be derived from H using rule 4, a contradiction.

Case 2: $u=0$ and H contains a downward $\exists$-literal. By repeatedly applying rule 4 to eliminate $\forall$-literals in H as many times as possible we obtain a new clause H'. If the last variable in H' (relative to the quantification order of ψ) is a universally quantified variable y, then for all $a\in D$, every extension of the assignment $y=a$ satisfies H', and so in particular H' is satisfied by f. On the other hand, if the last variable in H' is an $\exists$-variable x, then H' is an existential unit clause and is satisfied by τ_f by the definition of the τ_k.

Case 3 (Induction Step): Assume that H contains at least one upward $\exists$-literal, $x_k\not\leq_* d$, and that all derivable clauses with a smaller number of upward $\exists$-literals are satisfied by τ_f. Since clauses in ψ contain only non-trivial literals, it can be easily verified that H cannot contain the literal $x_k\not\leq_* 0$, that is, we have $d<_* 0$. Suppose (for contradiction) that τ_f does not satisfy H; then $\tau_f(x_k)\leq d$.

For $i=1,\dots,k$, denote $f(y_i)$ by a_i. Since $\tau_f(x_k)<_* 0$, there exists an existential unit clause C containing x_k such that $\psi\vdash_l C$ and C is not satisfied by the assignment $y_1=a_1,\dots,y_k=a_k,x_k=0$ (note that C is clearly satisfied by the assignment $y_1=a_1,\dots,y_k=a_k,x_k=1$). Consider the existential unit clause U obtained by applying rule 1 to all such clauses. Note that a further application of rule 1 to C and U would produce U again. Clearly, we have that $\psi\vdash_l U$ and U is not satisfied by the assignment $y_1=a_1,\dots,y_k=a_k,x_k=0$. Let $x_k\leq_* t$ be the only downward $\exists$-literal in U. We argue that $t=\tau_f(x_k)$. If, for some a, the assignment $y_1=a_1,\dots,y_k=a_k,x_k=a$ satisfies all derivable existential unit clauses containing x_k, then, in particular, it satisfies U. Since U is not satisfied by the assignment $y_1=a_1,\dots,y_k=a_k,x_k=0$, it follows that $a\leq_* t$. Then, by the definition of τ_f, we have $\tau_f(x_k)\leq_* t$. On the other hand, the assignment $y_1=a_1,\dots,y_k=a_k,x_k=t$ satisfies all derivable existential unit clauses containing x_k. Clearly, it satisfies all such clauses that are satisfied already by the assignment $y_1=a_1,\dots,y_k=a_k,x_k=0$. Furthermore, it satisfies the downward literal $x_k\leq_* t'$ in any other derivable existential unit clause C' containing x_k because, as we mentioned above, the application of rule 1 to C' and U gives U, implying that $t\circ t'=t$, which is equivalent to $t\leq_* t'$. Hence, by the definition of τ_f, we have $t\leq_*\tau_f(x_k)$, and so $t=\tau_f(x_k)$. To summarize, the derivable existential unit clause U contains the literal $x_k\leq_*\tau_f(x_k)$ and has the property that τ_f does not satisfy $U\setminus\{x_k\leq_*\tau_f(x_k)\}$.

Now by applying rule 2 we obtain $\psi\vdash_l U'$, where $U'=(U\setminus\{x_k\leq_*\tau_f(x_k)\})\cup\{x_k\leq d\}$. Finally, by applying rule 3 we obtain $\psi\vdash_l H'$, where $H'=(U'\setminus\{x_k\leq_* d\})\cup(H\setminus\{x_k\not\leq_* d\})$. Notice that H' is not satisfied by τ_f. Indeed, τ_f does not satisfy $U\setminus\{x_k\leq_*\tau_f(x_k)\}$, and $U'\setminus\{x_k\leq_* d\}$ is just the same clause. Furthermore, τ_f does not satisfy $H\setminus\{x_k\not\leq_* d\}$ because, by our assumption, it does not satisfy H. The clause H' contains one less upward $\exists$-literal than H, but is not satisfied by τ_f; this contradicts our inductive hypothesis. ∎

Proposition 6.5 *Let $*$ be an arbitrary semilattice operation with unit on a finite set D. For any finite $\Gamma \subseteq \mathsf{Inv}(\{*\})$, the problem class* QCSP(Γ) *is in* **PTIME**.

Proof: Let $\mathcal{P} = \forall y_1 \exists x_1 \ldots \forall y_n \exists x_n \psi$ be an instance of QCSP(Γ), and let $*$ be a semilattice operation with unit element 1 and zero element 0 under which Γ is invariant. By appeal to Lemma 6.2, every predicate invariant under $*$ can be represented as a conjunction of Horn-like clauses. In general, the size of this representation can grow exponentially (in the size of the predicate). However, for any fixed finite $\Gamma \subseteq \mathsf{Inv}(\{*\})$, this representation for all predicates in Γ can be found in constant time. Hence, we can assume that ψ contains only Horn-like clauses. We assume without loss of generality that no literals in ψ are trivial.

Let $B^{\exists}$ denote the subset of ψ containing all clauses with a downward $\exists$-literal. Let $B^{\forall}$ denote the subset of ψ containing all clauses with a downward $\forall$-literal; and let U denote the subset of ψ containing all clauses having only upward literals. It can be straightforwardly verified that for every clause C derivable from ψ by QCSP-literal-resolution, there is a single $H \in B^{\forall} \cup U$ such that C is derivable from $\mathcal{P}_H \stackrel{\text{def}}{=} \forall y_1 \exists x_1 \ldots \forall y_n \exists x_n (B^{\exists} \cup \{H\})$. Hence, by Lemma 6.4, deciding whether or not $\mathcal{P}$ is true amounts to deciding whether or not $\mathcal{P}_H$ is true, for all $H \in B^{\forall} \cup U$. We will therefore show how to decide any such $\mathcal{P}_H$ in polynomial time. There are two cases to consider:

Case 1: $H \in U$**.** In this case we claim that the sentence $\mathcal{P}_H$ is true if and only if the CSP instance

$$\exists x_1 \ldots \exists x_n \left(\bigwedge \{ C \setminus C^{\forall} \mid C \in B^{\exists} \cup \{H\} \} \right)$$

is satisfiable (where $C^{\forall}$ denotes the set of all $\forall$-literals in the clause C). To establish this claim note that if this CSP instance is satisfiable, a satisfying assignment for it gives a winning strategy for $\mathcal{P}_H$ (which ignores the $\forall$-player); on the other hand, if this CSP instance is unsatisfiable, then $\mathcal{P}_H$ is unsatisfiable as the $\forall$-player can set all $\forall$-variables to 1 to falsify all $\forall$-literals, causing the $\exists$-player to lose. Satisfiability of this CSP instance can be decided in polynomial time by the results of [35], because all predicates in it are invariant under a semilattice operation.

Case 2: $H \in B^{\forall}$**.** In this case, let y denote the variable in the downward literal of H and remove all $\forall$-literals not over y from the clauses of $B^{\exists} \cup \{H\}$ to obtain the set of clauses ψ'. We claim that $\mathcal{P}_H$ is true if and only if $\mathcal{P}'_H = \forall y_1 \exists x_1 \ldots \forall y_n \exists x_n (\psi')$ is true. This is because a QCSP-literal-resolution derivation of $\emptyset$ from $\mathcal{P}_H$ gives a derivation of $\emptyset$ from ψ' by removal of $\forall$-literals not including y; and, a derivation of $\emptyset$ from ψ' gives a derivation of $\emptyset$ from $\mathcal{P}_H$ by adding in $\forall$-literals as appropriate to derive (from $\mathcal{P}_H$) a clause consisting only of $\forall$-variables, from which $\emptyset$ can be derived by repeated applications of QCSP-literal-resolution rule 4 (with $a = 1$). Notice that when the last variable in a clause in the quantification order is a $\forall$-variable y, all literals involving y can be removed by using rule 4 with $a = 1$.

We now show how to decide if $\mathcal{P}'_H$ is true by reducing this question to an equivalent CSP instance. Let y_i denote the single $\forall$-variable occurring in ψ'. Clearly, $\mathcal{P}'_H$ is equivalent to $\mathcal{P}''_H = \exists x_1 \ldots \exists x_{i-1} \forall y_i \exists x_i \exists x_{i+1} \ldots \exists x_n (\psi')$. For all $a \in D$, define $\mathcal{P}''_a$ to be the formula $\mathcal{P}''_a = \exists x_1^a \ldots \exists x_n^a (\psi'[y_i/a]))$, that is, the formula obtained from $\mathcal{P}''_H$ by eliminating y_i from the quantifier prefix, instantiating y_i with the value a in ψ', and renaming each variable x_j as x_j^a. We want to find assignments satisfying the predicates of the $\mathcal{P}''_a$ such that, for each

$j = 1, \ldots, i-1$, the values received by the variables x_j^a are the same for all a. We can formulate the existence of such assignments as a CSP instance which has all the predicates of all the $\mathcal{P}''_a$ as constraints, as well as additional constraints $x_j^a = x_{j'}^a$ for all $a \in D$ and all $1 \le j < j' \le i-1$. This CSP instance is polynomial in the size of $\mathcal{P}_H$, and all predicates in it are invariant under the semilattice operation $*$. Hence, this instance can be decided in polynomial time by the results of [35].

■

6.2 Semilattice operations without unit

In this subsection, we show that if a semilattice operation $*$ on a set D has no unit then QCSP($\mathsf{Inv}(*)$) is **PSPACE**-complete.

First we note that if the semilattice operation $*$ has no unit then there are at least two different *minimal* elements with respect to $\le_*$. We shall fix two such elements a, b and denote the set $D \setminus \{a, b\}$ by E. Note that the minimality of a, b implies that, for any $d \in D$, if $d \ne a$ then $a * d \in E$, and if $d \ne b$ then $b * d \in E$.

To prove the **PSPACE**-completeness of QCSP($\mathsf{Inv}(*)$), we will make use of the following known combinatorial problem.

Definition 6.6 (SUCCINCT GRAPH UNREACHABILITY) *A* succinct representation *of a digraph with n vertices, where $n = 2^c$ is a power of two, is a Boolean circuit C with $2c$ inputs. The digraph represented by C, denoted G_C, is defined as follows: the vertices of G_C are $\{1, 2, \ldots, n\}$; the pair (i, j) is an edge of G_C if and only if C accepts the binary representations of the c-bit integers i, j as inputs.*

In the SUCCINCT GRAPH UNREACHABILITY *problem we are given a succinct representation of a digraph G and two vertices s, t of the graph. The question is whether there is no path in G that connects s and t.*

It is known (see, e.g., Exercise 20.2.9(b) of [41]) that the SUCCINCT GRAPH REACHABILITY problem is **PSPACE**-complete, and it follows that SUCCINCT GRAPH UNREACHABILITY is also **PSPACE**-complete.

Proposition 6.7 *Let $*$ be a semilattice operation without unit. The* SUCCINCT GRAPH UNREACHABILITY *problem is polynomial-time reducible to* QCSP(Γ) *where Γ is the set of all at most ternary relations from $\mathsf{Inv}(*)$.*

Proof: Let C be a succinct representation of a directed graph G_C. Encodings of vertices of G_C, that is c-tuples, will be denoted by $\vec{x}, \vec{y}$ etc., where $\vec{x} = (x_1, \ldots, x_c)$.

Let $\varrho_{\vec{s}}(\vec{x})$ denote the formula $\varrho_{s_1}(x_1) \wedge \ldots \wedge \varrho_{s_c}(x_c)$, where each ϱ_d is a *constant* relation, that is, a unary relation containing the single tuple (d). It is easily checked that each ϱ_d is invariant under the operation $*$.

Now define a predicate φ_C such that $\varphi_C(\vec{x}, \vec{y}, z_1, z_2)$ is true if and only if $\vec{x} = \vec{y}$, or there is a path in G_C from the vertex encoded $\vec{x}$ to the vertex encoded $\vec{y}$ in G_C and $z_1 = z_2$, or such a path does not exist, or one of $\vec{x}, \vec{y}$ does not belong to $\{a, b\}^c$. Note that $\varphi_C(\vec{x}, \vec{y}, z_1, z_2)$ is false precisely when $\vec{x}$ and $\vec{y}$ are encodings of distinct vertices in G_C which are connected by a path, and $z_1 \ne z_2$.

Using these predicates, an instance $C, \vec{s}, \vec{t}$ of the SUCCINCT GRAPH UNREACHABILITY problem can be reduced to the formula

$$\mathcal{P} = \exists \vec{x}, \vec{y}\ \forall z_1, z_2 \quad \varrho_{\vec{s}}(\vec{x}) \wedge \varrho_{\vec{t}}(\vec{y}) \wedge \varphi_C(\vec{x}, \vec{y}, z_1, z_2).$$

Hence there exists a polynomial-time reduction from SUCCINCT GRAPH UNREACHABILITY to QCSP(Γ) where Γ is the set of all at most ternary relations from $\mathsf{Inv}(*)$, provided that the predicate φ_C can be transformed to an instance of QCSP(Γ) in polynomial time.

Step 1: Expressing the predicate φ_C in simpler terms. We shall first show that the predicate φ_C can be expressed using the predicate $\varphi(\vec{x}, \vec{y}, z_1, z_2)$ which is defined as follows. Predicate $\varphi(\vec{x}, \vec{y}, z_1, z_2)$ is true if and only if $\vec{x}, \vec{y} \in \{a, b\}^c$ and $\vec{x} = \vec{y}$ or there is an edge in G_C from the vertex encoding $\vec{x}$ to the vertex encoding $\vec{y}$ in G_C and $z_1 = z_2$, or such an edge does not exist, or one of $\vec{x}, \vec{y}$ does not belong to $\{a, b\}^c$.

The most straightforward way to construct the predicate φ_C is to define it inductively, as follows:

$$\begin{aligned} \varphi'_0(\vec{x}, \vec{y}, z_1, z_2) &= \varphi(\vec{x}, \vec{y}, z_1, z_2), \\ \varphi'_i(\vec{x}, \vec{y}, z_1, z_2) &= \forall \vec{w}_i \exists z \;\; \varphi'_{i-1}(\vec{x}, \vec{w}_i, z_1, z) \wedge \varphi'_{i-1}(\vec{w}_i, \vec{y}, z, z_2). \end{aligned}$$

and $\varphi_C = \varphi'_c$. Unfortunately, φ'_c is exponentially larger than φ_C, so we cannot use this technique directly. However, we can use a standard trick to obtain a shorter expression for φ_C using universal quantifiers. To do this, we define the predicates φ_i inductively, as follows:

$$\begin{aligned} \varphi_0(\vec{x}, \vec{y}, z_1, z_2) &= \varphi(\vec{x}, \vec{y}, z_1, z_2), \\ \varphi_i(\vec{x}, \vec{y}, z_1, z_2) &= \forall \vec{w}_i \exists z \forall \vec{u}_i, \vec{v}_i \exists z'_1, z'_2 \\ &\qquad \varphi_{i-1}(\vec{u}_i, \vec{v}_i, z'_1, z'_2) \wedge \psi(\vec{x}, \vec{y}, \vec{u}, \vec{v}, \vec{w}, z, z_1, z_2, z'_1, z'_2). \end{aligned}$$

In the above expression, the predicate $\psi(\vec{x}, \vec{y}, \vec{u}, \vec{v}, \vec{w}, z, z_1, z_2, z'_1, z'_2)$ is defined by the following conditions:

- $\vec{x}, \vec{w}, \vec{u}, \vec{v} \in \{a, b\}^c$ and $\vec{u} = \vec{x}, \vec{v} = \vec{w}$ implies $z'_1 = z_1, z'_2 = z$, and
- $\vec{y}, \vec{w}, \vec{u}, \vec{v} \in \{a, b\}^c$ and $\vec{u} = \vec{w}, \vec{v} = \vec{y}$ implies $z'_1 = z, z'_2 = z_2$.

In other words, ψ is false only if the equalities on the left hold while the equalities on the right do not.

Finally, to obtain a QCSP instance we transform φ_c to prenex normal form moving all the quantifiers to the beginning of the formula preserving their order. It is not hard to see that the obtained formula is equivalent to φ_c. We set φ_C to be equal to this formula.

To show that this definition correctly captures φ_C, we prove by induction that $\varphi_i(\vec{x}, \vec{y}, z_1, z_2)$ is false precisely when $\vec{x}, \vec{y}$ are encodings of distinct vertices s, t of G_C which are connected by a path of length at most 2^i, but $z_1 \neq z_2$. The base case of induction follows from the definition of φ. Suppose that the result holds for φ_{i-1}.

Suppose first that $\vec{x}, \vec{y}$ are encodings of distinct vertices of G_C that are connected by a path of length at most 2^i. Choose some z_1, z_2. If $z_1 = z_2$ then take $z = z'_1 = z'_2 = z_1 = z_2$. In this case $\varphi_{i-1}(\vec{u}, \vec{v}, z'_1, z'_2)$ and $\psi(\vec{x}, \vec{y}, \vec{u}, \vec{v}, \vec{w}, z, z_1, z_2, z'_1, z'_2)$ hold for any $\vec{u}, \vec{v}, \vec{w}$, so $\varphi_i(\vec{x}, \vec{y}, z_1, z_2)$ is true. If $z_1 \neq z_2$ then, since $\vec{x}, \vec{y}$ are connected with a path of length at most 2^i, there is a vertex $\vec{w}$ such that $\vec{x}, \vec{w}$ and $\vec{w}, \vec{y}$ are connected with paths of length at most 2^{i-1}. Choose $\vec{u} = \vec{x}, \vec{v} = \vec{w}$. Then if $\varphi_{i-1}(\vec{u}, \vec{v}, z'_1, z'_2)$ is true then we have $z'_1 = z'_2$, and if $\psi(\vec{x}, \vec{y}, \vec{u}, \vec{v}, \vec{w}, z, z_1, z_2, z'_1, z'_2)$ is true then we have $z = z_1$. Similarly we can derive $z = z_2$, which means that $\varphi_i(\vec{x}, \vec{y}, z_1, z_2)$ is false.

Suppose now that $\vec{x}, \vec{y}$ are encodings of vertices of G_C that are *not* connected by a path of length at most 2^i. Choose some z_1, z_2. If $\vec{w} \notin \{a, b\}^c$ then set $z = z'_1 = z'_2 = a$. Under this assignment we have that $\psi(\vec{x}, \vec{y}, \vec{u}, \vec{v}, \vec{w}, z, z_1, z_2, z'_1, z'_2)$ and $\varphi_{i-1}(\vec{u}, \vec{v}, z'_1, z'_2)$ hold for any $\vec{u}, \vec{v}$, so $\varphi_i(\vec{x}, \vec{y}, z_1, z_2)$ is true. Hence we may assume that $\vec{w} \in \{a, b\}^c$. In this case at least one of the pairs $\vec{x}, \vec{w}$ and $\vec{w}, \vec{y}$

are not connected by a path of length at most 2^{i-1}. Without loss of generality suppose that there is no such path for $\vec{x},\vec{w}$. Then set $z = z_2$. If neither $\vec{u} = \vec{x}, \vec{v} = \vec{w}$ nor $\vec{u} = \vec{w}, \vec{v} = \vec{y}$ then by setting $z'_1 = z'_2 = a$ we make both predicates true, so $\varphi_i(\vec{x},\vec{y}, z_1, z_2)$ is true. If $\vec{u} = \vec{x}, \vec{v} = \vec{w}$ then we set $z'_1 = z_1, z'_2 = z$. Then $\psi(\vec{x},\vec{y},\vec{u},\vec{v},\vec{w}, z, z_1, z_2, z'_1, z'_2)$ is true. Since $\vec{u},\vec{v}$ are not connected with a path of length 2^{i-1}, $\varphi_{i-1}(\vec{u},\vec{v}, z'_1, z'_2)$ is also true. Finally, if $\vec{u} = \vec{w}, \vec{v} = \vec{y}$ we set $z'_1 = z, z'_2 = z_2$, and, as $z = z_2$, both predicates are true, so again $\varphi_i(\vec{x},\vec{y}, z_1, z_2)$ is true.

Finally, suppose that one of $\vec{x},\vec{y}$ does not belong to $\{a, b\}^c$, say, $\vec{x} \notin \{a, b\}^c$. If $\vec{w} \notin \{a, b\}^c$ then we proceed as above. Otherwise set $z = z_2$. If $\vec{u} \neq \vec{w}$ or $\vec{v} \neq \vec{y}$ then setting $z'_1 = z'_2 = a$ we make both predicates true. If $\vec{u} = \vec{w}$ and $\vec{v} = \vec{y}$ then set $z'_1 = z, z'_2 = z_2$.

This completes the proof by induction and establishes that the predicate φ_C can be transformed in polynomial time into a QCSP instance containing only the predicates φ and ψ. As is easily seen both predicates, φ and ψ, are invariant under the semilattice operation, however, they do not fit our purpose, because the explicit representations of these predicates are exponential in the size of the original Succinct Graph Unreachability instance. Thus, we need to show that these two predicates can be expressed by using at most ternary predicates from $\mathsf{Inv}(*)$ in polynomial time.

Step 2: Expressing the predicates φ and ψ. First, we introduce three relations corresponding to three types of logic gates. We will call these relations *gate relations*. The relations are partly given by their matrices (where columns correspond to tuples); the initial block of tuples contains the tuples that encode the gate, while the remaining tuples are needed for technical purposes and to obtain a relation invariant under the semilattice operation. Element a will be interpreted as FALSE and b as TRUE.

$$\begin{aligned}
\varrho_{\mathrm{NOT}} &= \begin{pmatrix} a & b \\ b & a \end{pmatrix} \cup (E \times D) \\
\varrho_{\mathrm{OR}} &= \begin{pmatrix} a & a & b & b \\ a & b & a & b \\ a & b & b & b \end{pmatrix} \cup (E \times D \times D) \cup (D \times E \times D) \\
\varrho_{\mathrm{AND}} &= \begin{pmatrix} a & a & b & b \\ a & b & a & b \\ a & a & a & b \end{pmatrix} \cup (E \times D \times D) \cup (D \times E \times D)
\end{aligned}$$

It is straightforward to verify that each of these gate relations is invariant under the semilattice operation $*$.

The circuit C representing the graph G_C is a Boolean circuit with gates $\{g_1, \ldots, g_k\}$, inputs $u_1, \ldots, u_\ell$ and output z. For each gate g_i, we denote the inputs of g_i by x_i, y_i (to simplify the notation we shall assume that if g_i is a NOT-gate then it still has the second input y_i, but it is void), and its output by z_i. Then $u_1, \ldots, u_\ell \in \{x_1, \ldots, x_k, y_1, \ldots, y_k\}$, $z \in \{z_1, \ldots, z_k\}$ and $z_1, \ldots, z_k \in \{x_1, \ldots, x_k, y_1, \ldots, y_k\} \cup \{z\}$. Without loss of generality we may assume that $z = z_k$. We will also assume that C has no unused inputs, that is, in the graph representation of C there is a path from every u_i to z. The encoding of circuit C will be the following existential conjunctive formula:

$$\theta_C(u_1, \ldots, u_\ell, z) = \exists z_1, \ldots, z_{k-1} \bigwedge_{i=1}^{k} \varrho_{w_i}(x_i, y_i, z_i),$$

where w_i denotes the type of gate g_i: NOT, AND, or OR.

We need three observations about the formula θ_C.

(1) If $u_1, \ldots, u_\ell \in \{a, b\}$ and the quantifier free part of $\theta_C(u_1, \ldots, u_\ell, z)$ is satisfied, then $z_1, \ldots, z_k \in \{a, b\}$.

This is easily verified using induction on the depth of circuit C, and the fact that, for any $w \in \{\text{NOT,AND,OR}\}$, if $x, y \in \{a, b\}$ and $\varrho_w(x, y, z)$ holds then $z \in \{a, b\}$.

(2) If $u_1, \ldots, u_\ell \in \{a, b\}$ and $\theta_C(u_1, \ldots, u_\ell, z)$ holds, then $z = b$ if and only if $C(u_1, \ldots, u_\ell)$ is TRUE; otherwise $z = a$.

Again this is easily verified using induction on the depth of the circuit and the structure of the relations.

(3) If $\{u_1, \ldots, u_\ell\} \cap E \neq \emptyset$ then $\theta_C(u_1, \ldots, u_\ell, z)$ holds for any $z \in D$.

To establish this, assume without loss of generality that $u_j \in E$. We proceed by induction on the depth of circuit C. Recall that C is assumed to have no unused inputs. In the base case of induction, when C contains only one gate, the result follows from the definitions of the gate relations. For the induction step, remove the output gate g_k from C; the rest of C then breaks into two circuits C_1 and C_2 (which may overlap). If g_k is a NOT-gate C_2 can be assumed to be empty. At least one of them uses input u_j; without loss of generality we assume it is C_1. Let the output of C_1 be z_{k-1}. By the induction hypothesis, $\theta_{C_1}(u_1, \ldots, u_\ell, d)$ holds for any $d \in E$. Since z_{k-1} is an input for g_k, the result follows from the definition of the gate relations.

Now let η be the following ternary relation

$$\eta = \{ \begin{pmatrix} b \\ d \\ d \end{pmatrix} \mid d \in D\} \cup ((\{a\} \cup E) \times D \times D).$$

It is straightforward to verify that η is invariant under the semilattice operation $*$.

We claim that the predicate φ can be expressed in terms of the predicates θ_C and η in the following way:

$$\varphi(\vec{x}, \vec{y}, z_1, z_2) = \exists z \;\; \theta_C(\vec{x}, \vec{y}, z) \wedge \eta(z, z_1, z_2).$$

To establish this claim, note first that if either $\vec{x}$ or $\vec{y}$ contains a component from E, then choosing $z = a$ we satisfy both predicates on the right-hand side. The same is true if $C(\vec{x}, \vec{y})$ is FALSE. Finally, if $\vec{x}, \vec{y}$ correspond to vertices that are connected, that is, $C(\vec{x}, \vec{y})$ is TRUE, then the only value of z satisfying θ_C is b, so to satisfy η we have to have $z_1 = z_2$.

We have shown that the predicate φ can be expressed in polynomial time by using at most ternary predicates from $\mathsf{Inv}(*)$. It only remains to show that the predicate ψ defined earlier can also be expressed in polynomial time by using at most ternary predicates from $\mathsf{Inv}(*)$.

Define the relation σ, as follows

$$\sigma = \begin{pmatrix} a & b \\ a & b \\ b & b \end{pmatrix} \cup (E \times E \times \{b\}) \cup (((D \times D) \setminus \{(a, a), (b, b)\}) \times (\{a\} \cup E)).$$

It is straightforward to verify that σ is invariant under the semilattice operation $*$. Let

$$\xi_{=}(\vec{x}, \vec{y}, z) \;\; = \;\; \exists s_1, \ldots, s_c \exists z_1, \ldots, z_{c-2} \bigwedge_{i=1}^{c} \sigma(x_i, y_i, s_i) \wedge \varrho_{\text{AND}}(s_1, s_2, z_1)$$
$$\wedge \bigwedge_{i=2}^{c-2} \varrho_{\text{AND}}(z_{i-1}, s_{i+1}, z_i) \wedge \varrho_{\text{AND}}(z_{c-2}, s_c, z).$$

Furthermore, let

$$\xi_{\to}(\vec{x},\vec{y},\vec{u},\vec{v},z_1,z_2,z'_1,z'_2) = \exists s_1,s_2\exists t \\ \xi_{=}(\vec{x},\vec{u},s_1)\wedge\xi_{=}(\vec{y},\vec{v},s_2)\wedge\varrho_{\mathrm{AND}}(s_1,s_2,t)\wedge\eta(t,z_1,z'_1)\wedge\eta(t,z_2,z'_2).$$

We claim that

$$\psi(\vec{x},\vec{y},\vec{u},\vec{v},\vec{w},z,z_1,z_2,z'_1,z'_2) \quad = \quad \xi_{\to}(\vec{x},\vec{w},\vec{u},\vec{v},z_1,z,z'_1,z'_2) \quad \wedge \\ \xi_{\to}(\vec{w},\vec{y},\vec{u},\vec{v},z,z_2,z'_1,z'_2).$$

To establish this claim, first consider the predicate $\xi_{=}(\vec{x},\vec{y},z)$. If this predicate holds and $\vec{x}=\vec{y}$ then all the s_i equal b, and furthermore all the z_i and z equal b. If $\vec{x}\neq\vec{y}$ or one of them does not belong to $\{a,b\}^c$ then, for some i, we have $x_i\neq y_i$ or $\{x_i,y_i\}\cap E\neq\emptyset$, and since $\sigma(x_i,y_i,s_i)$ is true, s_i can be chosen from E. Therefore, $z_{i-1},\ldots,z_{c-2},z$ can be chosen arbitrarily. Thus, $\xi_{=}(\vec{x},\vec{y},z)$ is true if and only if either $\vec{x}=\vec{y}$ and $\vec{x},\vec{y}\in\{a,b\}^c$ and $z=b$, or else $\vec{x}\neq\vec{y}$ and z is arbitrary, or $\vec{x},\vec{y}\notin\{a,b\}^c$ and z is arbitrary. Similarly, the predicate $\xi_{\to}(\vec{x},\vec{y},\vec{u},\vec{v},z_1,z_2,z'_1,z'_2)$ is true if and only if either one of $\vec{x},\vec{y},\vec{u},\vec{v}$ is not a member of $\{a,b\}^c$, or $\vec{x}\neq\vec{u}$, or $\vec{y}\neq\vec{v}$, or $\vec{x}=\vec{u},\vec{y}=\vec{v}$ and $z_1=z'_1, z_2=z'_2$.

Now consider $\psi(\vec{x},\vec{y},\vec{u},\vec{v},\vec{w},z,z_1,z_2,z'_1,z'_2)$. Suppose that $\vec{x},\vec{w},\vec{u},\vec{v}\in\{a,b\}^c$, $\vec{u}=\vec{x},\vec{v}=\vec{w}$. Then, to ensure that $\xi_{\to}(\vec{x},\vec{w},\vec{u},\vec{v},z_1,z,z'_1,z'_2)=1$, we must have $z'_1=z_1,z'_2=z$. Similarly, if $\vec{u}=\vec{w},\vec{v}=\vec{y}$ then $z'_1=z,z'_2=z_2$. If these equalities do not hold, or one of $\vec{x},\vec{w}$ and one of $\vec{w},\vec{y}$ are not members of $\{a,b\}^c$, then both predicates $\xi_{\to}$ are true for any z,z_1,z_2,z'_1,z'_2 and so is ψ.

As easily seen, the number of predicates used to represent φ, ψ and φ_C is bounded by a linear polynomial in the number of gates in circuit C. Therefore, the construction described is a polynomial-time reduction from the SUCCINCT GRAPH UNREACHABILITY problem to QCSP(Γ) where Γ is the set of at most ternary predicates invariant under the semilattice operation. ■

7 A Trichotomy Theorem

In this section we apply results from the previous sections to obtain a complete classification of complexity of QCSP(Γ) in those cases where Γ contains the set Δ of all graphs of permutations. Recall that the graph of a permutation π is the binary relation $\{(x,y)\mid y=\pi(x)\}$ (or the binary predicate $\pi(x)=y$). The complexity of CSP(Γ) for such sets Γ is completely classified in [22].

We will need two new surjective operations:

- The k-ary *near projection* operation,

$$l_k(x_1,\ldots,x_k)=\begin{cases} x_1 & \text{if } x_1,\ldots,x_k \text{ are all different,}\\ x_k & \text{otherwise.}\end{cases}$$

- The ternary *switching* operation,

$$s(x,y,z)=\begin{cases} x & \text{if } y=z,\\ y & \text{if } x=z,\\ z & \text{otherwise.}\end{cases}$$

Recall that the *dual discriminator* operation is defined as follows:

$$d(x,y,z) = \begin{cases} y & \text{if } y = z \\ x & \text{otherwise.} \end{cases}$$

Proposition 7.1 *If* $\Gamma \subseteq R_D$, $|D| \geq 3$, *and* $l_{|D|} \in \mathsf{s\text{-}Pol}(\Gamma)$ *then* $\mathrm{QCSP}(\Gamma)$ *is polynomial-time reducible to* $\mathrm{CSP}(\Gamma')$ *where* $\Gamma' = \mathsf{Inv}(\mathsf{Pol}(\Gamma))$. *In particular,* $\mathrm{QCSP}(\Gamma)$ *is in* **NP**.

Proof: For any $\vec{a} = (a_1, \ldots, a_n) \in D^n$, and any subsequence $i_1, \ldots, i_k$ of the sequence $1, \ldots, n$, we define $\mathrm{pr}_{i_1,\ldots,i_k}\vec{a}$ to be the k-tuple $(a_{i_1}, \ldots, a_{i_k})$. Moreover, for any n-ary relation ϱ, we define $\mathrm{pr}_{i_1,\ldots,i_k}\varrho$ to be the k-ary relation

$$\mathrm{pr}_{i_1,\ldots,i_k}\varrho = \{\mathrm{pr}_{i_1,\ldots,i_k}\vec{a} \mid \vec{a} = (a_1, \ldots, a_n) \in \varrho\}.$$

For $I = \{i_1, \ldots, i_k\}$, we will sometimes write $\mathrm{pr}_I\varrho$ instead of $\mathrm{pr}_{i_1,\ldots,i_k}\varrho$. Note that $\mathsf{Pol}(\{\mathrm{pr}_I\varrho\}) \supseteq \mathsf{Pol}(\{\varrho\})$.

We first clarify the structure of relations over a set D which are invariant under the near-projection operation $l_{|D|}$.

Let $\underline{n}$ denote the set $\{1, \ldots, n\}$. Suppose $I_1, \ldots, I_k$ is a partition of $\underline{n}$ and let $\varrho_j = \mathrm{pr}_{I_j}\varrho$ for $j = 1, \ldots, k$. Then we write $\varrho = \varrho_1 \times \ldots \times \varrho_k$ if ϱ can be represented as $\varrho = \{\vec{a} \mid \mathrm{pr}_{I_j}\vec{a} \in \varrho_j \text{ for every } j = 1, \ldots, k\}$.

Lemma 7.2 *Let* $\varrho \in R_D^{(n)}$, *where* $|D| \geq 3$.

If $\varrho \in \mathsf{Inv}(\{l_{|D|}\})$ *and* $\mathrm{pr}_i\varrho = D$ *for every* $i \in \underline{n}$, *then* ϱ *is of the form*

$$\varrho = \varrho_1 \times \ldots \times \varrho_k$$

where each $\varrho_j = \{(a, \pi_{2j}(a), \ldots, \pi_{m_j j}(a)) \mid a \in D\}$, *for some permutations* $\pi_{2j}, \ldots, \pi_{m_j j}$ *of* D.

Proof: We prove the lemma by induction on n. When $n = 1$ the result holds trivially, so consider the case when $n = 2$. Assume that ϱ is not a graph of a permutation. Then there exist $b_1, b_2, b \in D$ such that $b_1 \neq b_2$ and $(b_1, b), (b_2, b) \in \varrho$ (or $(b, b_1), (b, b_2) \in \varrho$). Since $\mathrm{pr}_1\varrho = D$, it is possible to choose $\vec{a}_1, \vec{a}_2, \ldots, \vec{a}_{|D|} \in \varrho$ so that $\mathrm{pr}_1\vec{a}_1, \ldots, \mathrm{pr}_1\vec{a}_{|D|}$ are all different, $\mathrm{pr}_1\vec{a}_1 = (x)$, where x is an arbitrary element from $D \setminus \{b_1, b_2\}$, $\vec{a}_2 = (b_1, b)$, and $\vec{a}_{|D|} = (b_2, b)$. Since ϱ is invariant under $l_{|D|}$, we have $l_{|D|}(\vec{a}_1, \ldots, \vec{a}_k) = (\mathrm{pr}_1\vec{a}_1, b) \in \varrho$, and hence $(x, b) \in \varrho$ for all $x \in D$.

It follows that, for any $(x, y) \in D^2$ we can choose $\vec{c}_1, \vec{c}_2, \ldots, \vec{c}_{|D|} \in \varrho$ such that $\mathrm{pr}_1\vec{c}_1, \ldots, \mathrm{pr}_1\vec{c}_{|D|}$ are all different, $\mathrm{pr}_1\vec{c}_1 = (x)$, $\mathrm{pr}_2\vec{c}_{|D|} = (y)$, and $\mathrm{pr}_2\vec{c}_1 = \ldots = \mathrm{pr}_2\vec{c}_{|D|-1} = (b)$. Since ϱ is invariant under $l_{|D|}$, we have $l_{|D|}(\vec{c}_1, \ldots, \vec{c}_{|D|}) = (x, y) \in \varrho$, and hence $\varrho = D^2$.

We now prove the induction step. By the argument above, for any pair $i, j \in \underline{n}$ the projection $\mathrm{pr}_{i,j}\varrho$ is either D^2, or the graph of a permutation. Assume that there exist i, j such that $\mathrm{pr}_{i,j}\varrho$ is the graph of a permutation π. By the inductive hypothesis $\mathrm{pr}_{\underline{n}\setminus\{j\}}\varrho$ can be represented in the form

$$\mathrm{pr}_{\underline{n}\setminus\{j\}}\varrho = \varrho_1 \times \ldots \times \varrho_k,$$

and the i-th coordinate position occurs in one of $\varrho_1, \ldots, \varrho_k$. Suppose, for simplicity, that i is the last coordinate position in ϱ_1, that is,

$$\begin{aligned} \varrho_1 \;\; = \;\; & \{(a_{i_1}, \ldots, a_{i_{m_1-1}}, a_i) \mid a_{i_1} \in D,\ a_{i_s} = \pi_{s1}(a_{i_1}) \\ & \text{for } s \in \{2, \ldots, m_1 - 1\},\ a_i = \pi_i(a_{i_1})\}. \end{aligned}$$

Then, letting

$$\begin{array}{rcl}\varrho_1' & = & \{(a_{i_1},\ldots,a_{i_{m_1-1}},a_i,a_j) \mid a_{i_1}\in D,\ a_{i_s}=\pi_{s1}(a_{i_1}) \\ & & \quad \text{for } s\in\{2,\ldots,m_1-1\},\ a_i=\pi_i(a_{i_1}),a_j=\pi\pi_i(a_{i_1})\}\end{array}$$

we have $\varrho = \varrho_1' \times \varrho_2 \times \ldots \times \varrho_k$, as required.

It remains to prove that if $\mathrm{pr}_{i,j}\varrho = D^2$ for every $i,j \in \underline{n}$, then $\varrho = D^n$.

For any $a \in D$, define

$$\varrho_a = \{(a_1,\ldots,a_{n-1}) \mid (a_1,\ldots,a_{n-1},a) \in \varrho\}.$$

Since the operation $l_{|D|}$ is idempotent, that is, it satisfies $l_{|D|}(x,\ldots,x) = x$ for all x, ϱ_a also belongs to $\mathsf{Inv}(\{l_{|D|}\})$.

Consider first the case $n = 3$.

Suppose that, for some $a \in D$, the relation ϱ_a is not the graph of a permutation. Then $\varrho_a = D^2$ by the argument above. Take any $c \in D$ such that $c \neq a$. Then there exists a tuple $\vec{c} = (c_1, c_2, c) \in \varrho$. For $1 \leq i \leq |D|-1$, choose $\vec{a}_i = (x_i, y_i, a) \in \varrho$, such that $\{x_1,\ldots,x_{|D|-1},c_1\} = \{y_1,\ldots,y_{|D|-1},c_2\} = D$. Then $l_{|D|}(\vec{a}_1,\ldots,\vec{a}_{|D|-1},\vec{c}) = (x_1,y_1,c) \in \varrho$. Since we can change y_1 whilst keeping the same x_1, we conclude that ϱ_c is not the graph of a permutation. Thus $\varrho_c = D^2$ for all $c \in D$, which implies that $\varrho = D^3$.

Now consider the remaining case, where ϱ_a is the graph of a permutation for each $a \in D$. In this case $|\varrho_a| = |D|$ for each a, and since $|\mathrm{pr}_{1,2}\varrho| = |\bigcup_{a\in D}\varrho_a| = |D^2|$, we have $\varrho_a \cap \varrho_b = \emptyset$ for all $a,b \in D$ such that $a \neq b$. Assume that $D = \{d_1,d_2,\ldots,d_{|D|}\}$. For $1 \leq i \leq |D|-1$, choose $\vec{a}_i = (a_i,d_1,d_i) \in \varrho$, and choose $\vec{a}_{|D|} = (a_1,b,d_{|D|}) \in \varrho$. Note that $a_1,\ldots,a_{|D|-1}$ are all different, and $b \neq d_1$, because $\varrho_{d_i} \cap \varrho_{d_j} = \emptyset$ if $i \neq j$. Now $l_{|D|}(\vec{a}_1,\ldots,\vec{a}_{|D|}) = (a_1,b,d_1) \in \varrho$, so $(a_1,b) \in \varrho_{d_1} \cap \varrho_{d_{|D|}}$, a contradiction.

If $n > 3$ then, by the inductive hypothesis, we have $\mathrm{pr}_{i,j,n}\varrho = D^3$ and, consequently, $\mathrm{pr}_{i,j}\varrho_a = D^2$ holds for all $1 \leq i,j \leq n-1$. Applying the inductive hypothesis to ϱ_a, we obtain $\varrho_a = D^{n-1}$ for each $a \in D$, which implies that $\varrho = D^n$. ∎

Lemma 7.3 *Let $\varrho \in R_D^{(n)}$, where $|D| \geq 3$.*
If $\varrho \in \mathsf{Inv}(\{l_{|D|}\})$ and $I = \{i \in \underline{n} \mid |\mathrm{pr}_i\varrho| < |D|\}$, then $\varrho = \mathrm{pr}_I\varrho \times \mathrm{pr}_{\underline{n}\setminus I}\varrho$.

Proof: By Lemma 7.2, $\mathrm{pr}_{\underline{n}\setminus I}\varrho = \varrho_1 \times \ldots \times \varrho_k$ where $\varrho_j = \{(a,\pi_{2j}(a),\ldots,\pi_{m_jj}(a)) \mid a \in D\}$ and $\pi_{2j},\ldots,\pi_{m_jj}$ are permutations of D. Denote the set of coordinate indices of ϱ_j by J_j, and let J be a system of representatives of $J_1,\ldots,J_k$. Then $\mathrm{pr}_J\varrho = D^{|J|}$.

Take an arbitrary $\vec{b} \in \mathrm{pr}_I\varrho$ and $\vec{c} \in \mathrm{pr}_J\varrho$. There exists $\vec{a}_{|D|} \in \mathrm{pr}_{I\cup J}\varrho$ such that $\mathrm{pr}_I\vec{a}_{|D|} = \vec{b}$. For $1 \leq i \leq |D|-1$, choose $\vec{a}_i \in \mathrm{pr}_{I\cup J}\varrho$ such that $\mathrm{pr}_J\vec{a}_1 = \vec{c}$ and for each $j \in J$, $\{\vec{a}_i(j) \mid 1 \leq i \leq |D|\} = D$. (This is possible because $\mathrm{pr}_J\varrho = D^{|J|}$.)

Now let $\vec{d} = l_{|D|}(\vec{a}_1,\ldots,\vec{a}_{|D|}) \in \mathrm{pr}_{I\cup J}\varrho$. It is easy to check that $\mathrm{pr}_I\vec{d} = \vec{b}$ and $\mathrm{pr}_J\vec{d} = \vec{c}$. Hence, $\mathrm{pr}_{I\cup J}\varrho = \mathrm{pr}_I\varrho \times \mathrm{pr}_J\varrho$. Finally, by the choice of J, any element from $\mathrm{pr}_{I\cup J}\varrho$ has a unique extension to an element of ϱ, and the result follows. ∎

Now we can describe a polynomial-time reduction from QCSP(Γ) to CSP($\mathsf{Inv}(\mathsf{Pol}(\Gamma))$).

Let $\mathcal{P} = \mathcal{Q}_1x_1\ldots\mathcal{Q}_lx_l\ \phi$ be an instance of QCSP(Γ) where $\phi = \varrho_1 \wedge \ldots \wedge \varrho_q$. Let ϕ' be a CSP instance obtained from ϕ by establishing 2-consistency (see Definition 4.3). As explained in Section 4.2, ϕ' can be obtained from ϕ in polynomial time, and has the same set of satisfying assignments. Moreover, all predicates in ϕ' belong to $\langle\Gamma\rangle$, and hence, by Proposition 3.3, to $\mathsf{Inv}(\mathsf{Pol}(\Gamma))$.

Let $P' = \mathcal{Q}_1 x_1 \dots \mathcal{Q}_l x_l\ \phi'$. Note that, since ϕ' is 2-consistent, if two constraints in ϕ' share a variable then the projections of the corresponding predicates on the coordinates where this variable occurs are the same. If one of the predicates in ϕ' is always false, or if some variable that cannot take all values is universally quantified in $\mathcal{P}'$, then clearly $\mathcal{P}'$ is false, and so is $\mathcal{P}$. Otherwise, by Lemmas 7.2 and 7.3, ϕ' can be represented as $\phi_1 \wedge \phi_2$ so that ϕ_1 and ϕ_2 have no variable in common, ϕ_1 is a conjunction of graphs of permutations, and no variable in ϕ_2 can take all possible values. Hence $\mathcal{P}'$ can be represented as a conjunction of two sentences: one (corresponding to ϕ_1) is an instance of QCSP(Δ), and the other is an instance of CSP($\mathsf{Inv}(\mathsf{Pol}(\Gamma))$). It is easy to check that $\Delta \subseteq \mathsf{Inv}(d)$. Hence, since the dual discriminator operation is a near-unanimity operation, by Theorem 4.5, we can solve any instance of QCSP(Δ) in polynomial time, and we have reduced QCSP(Γ) to CSP($\mathsf{Inv}(\mathsf{Pol}(\Gamma))$). ■

Theorem 7.4 *Let $\Delta \subseteq \Gamma \subseteq R_D$, and $|D| \geq 3$.*

- *If* $\mathsf{s\text{-}Pol}(\Gamma)$ *contains the dual discriminator d, or the switching operation s, or an affine operation, then* QCSP(Γ) *is in* **PTIME***;*
- *else, if* $\mathsf{s\text{-}Pol}(\Gamma)$ *contains $l_{|D|}$, then* QCSP(Γ) *is* **NP***-complete;*
- *else* QCSP(Γ) *is* **PSPACE***-complete.*

Proof: Chapter 5 of [50] shows that, either $\mathsf{s\text{-}Pol}(\Gamma)$ consists of all projections (that is, all functions of the form $f(x_1, \dots, x_n) = x_i$ for some $1 \leq i \leq n$), or else $\mathsf{s\text{-}Pol}(\Gamma)$ contains the dual discriminator operation, d, or the near-projection operation, $l_{|D|}$, or (when $|D| \in \{3, 4\}$) an affine operation. If $\mathsf{s\text{-}Pol}(\Gamma)$ consists of all projections then, by Theorem 5.2, QCSP(Γ) is **PSPACE**-complete. If $\mathsf{s\text{-}Pol}(\Gamma)$ contains d or an affine operation then, by Theorem 4.2 or Theorem 4.5, QCSP(Γ) is in **PTIME**.

Suppose that $\mathsf{s\text{-}Pol}(\Gamma)$ contains $l_{|D|}$, but neither d nor the affine operation. Then, by Proposition 7.1, QCSP(Γ) is in **NP**. Note that s is a Mal'tsev operation, and, hence, if $\mathsf{s\text{-}Pol}(\Gamma)$ contains s then QCSP(Γ) is solvable in polynomial time by Theorems 4.2. If $\mathsf{s\text{-}Pol}(\Gamma)$ contains none of s, d, and the affine operation then, by Theorem 12 of [22], CSP(Γ) is **NP**-complete. Since, obviously, CSP(Γ) is polynomial-time reducible to QCSP(Γ), and QCSP(Γ) is in **NP**, the result follows. ■

Note that, for any fixed finite set D, the conditions in Theorem 7.4 can be efficiently checked.

8 Conclusions

We have shown that the algebraic theory relating complexity and polymorphisms, which was originally developed for the standard constraint satisfaction problem allowing only existential quantifiers, can be extended to deal with the more general framework of the quantified constraint satisfaction problem.

In this extension of the theory it turns out that it is the *surjective* polymorphisms of the predicates used in problem instances which determine the complexity of the corresponding problems. Using this information we have been able to identify subproblems of the quantified constraint satisfaction problem lying in (or complete for) some standard complexity classes.

As an example of using these results, we now classify the complexity of the constraint satisfaction games described in Examples 2.7 to 2.10.

Corollary 8.1

(1) The Graph k-Colouring Game *described in Example 2.7 can be decided in polynomial time when* $k \leq 2$ *and is* **PSPACE**-*complete when* $k \geq 3$.

(2) The One-Or-Both Colour Matching Game *described in Example 2.8 can be decided in polynomial time.*

(3) The Colour Implication Game *described in Example 2.9 can be decided in polynomial time.*

(4) The Linear Equations Game *described in Example 2.10 can be decided in polynomial time.*

Proof:

(1) Follows immediately from Corollary 3.9 and Proposition 5.1.

(2) It is straightforward to verify that each relation in Γ_{cm} defined in Example 2.8 is invariant under the dual discriminator operation, which is a near unanimity operation. Hence, by Theorem 4.5, the One-Or-Both Colour Matching Game can be decided in polynomial time.

(3) The Colour Implication Game defined in Example 2.9 involves a set D of colours containing two distinguished colours, black and white. Consider the binary operation $*$ on D such that, for all $v \in D$, we have $v * v = v * \text{black} = \text{black} * v = v$, and, for any distinct $u, v \in D \setminus \{\text{black}\}$, we have $u * v = \text{white}$. It is easy to check that $*$ is a semilattice operation where the black colour is a unit element. The corresponding lattice order $\leq_*$ is a so-called "diamond" order: it has black as the least element, white as the greatest element, and all other colours incomparable with each other. It is straightforward to verify that each of the relations $\varrho_{a,b}$ defined in Example 2.9 is equal to the set $\{(u, v) \mid (u \not\leq_* a) \vee (v \leq_* b)\}$. Hence, by Lemma 6.2, the set of relations Γ_{ci} is invariant under the operation $*$. The result then follows from Theorem 6.1.

(4) It is straightforward to verify that each relation in Γ_{lin} defined in Example 2.10 is invariant under the affine operation of the field K. Hence, by Theorem 4.2, the Linear Equations Game can be decided in polynomial time.

■

Acknowledgements

Part of this work was done when A. Bulatov, H. Chen, and A. Krokhin visited the Isaac Newton Institute for Mathematical Sciences in Cambridge, UK. Financial support from the Institute is gratefully acknowledged.

References

[1] B. Aspvall, M.F. Plass, and R.E. Tarjan. A linear time algorithm for testing the truth of certain quantified Boolean formulas. *Information Processing Letters*, 8:121–123, 1979.

[2] H.L. Bodlaender. On the complexity of some coloring games. *International Journal of Foundations of Computer Science*, 2(2):133–147, 1991.

[3] L. Bordeaux and E. Monfroy. Beyond NP: Arc-consistency for quantified constraints. In *Proceedings of CP'02*, volume 2470 of *LNCS*, pages 371–386, 2002.

[4] F. Börner, A. Bulatov, P. Jeavons, and A. Krokhin. Quantified constraints: Algorithms and complexity. In *Proceedings of CSL'03*, volume 2803 of *LNCS*, pages 58–70, 2003.

[5] F. Börner, A. Krokhin, A. Bulatov, and P. Jeavons. Quantified constraints and surjective polymorphisms. Technical Report PRG-RR-02-11, Computing Laboratory, University of Oxford, UK, 2002.

[6] A. Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *Journal of the ACM*, 53(1):66-120, 2006.

[7] A. Bulatov. Tractable conservative constraint satisfaction problems. In *Proceedings of LICS'03*, pages 321–330, 2003.

[8] A. Bulatov. A graph of a relational structure and constraint satisfaction problems. In *Proceedings of LICS'04*, pages 448–457, 2004.

[9] A. Bulatov. Combinatorial problems raised from 2-semilattices. *Journal of Algebra*, 298(2):321-339, 2006.

[10] A. Bulatov and V. Dalmau. A simple algorithm for Mal'tsev constraints. *SIAM Journal on Computing*, 36(1):16-27, 2006.

[11] A. Bulatov and P. Jeavons. An algebraic approach to multi-sorted constraints. In *Proceedings of CP'03*, volume 2833 of *LNCS*, pages 183–198, 2003.

[12] A. Bulatov, P. Jeavons, and A. Krokhin. Classifying the complexity of constraints using finite algebras. *SIAM Journal on Computing*, 34(3):720–742, 2005.

[13] M. Cadoli, M. Schaerf, A. Giovanardi, and M. Giovanardi. An algorithm to evaluate quantified Boolean formulae and its experimental evaluation. *Journal of Automated Reasoning*, 28(2):101–142, 2002.

[14] H. Chen. Collapsibility and consistency in quantified constraint satisfaction. In *Proceedings of AAAI'04*, pages 155–160, 2004.

[15] H. Chen. *The Computational Complexity of Quantified Constraint Satisfaction.* PhD thesis, Cornell University, 2004.

[16] H. Chen. Quantified constraint satisfaction and 2-semilattice polymorphisms. In *Proceedings of CP'04*, volume 3258 of *LNCS*, pages 168–181, 2004.

[17] H. Chen. Quantified constraint satisfaction, maximal constraint languages, and symmetric polymorphisms. In *Proceedings of STACS'05*, volume 3404 of *LNCS*, pages 315–326, 2005.

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

[19] V. Dalmau. Some dichotomy theorems on constant-free quantified Boolean formulas. Technical Report TR LSI-97-43-R, Department LSI, Universitat Politecnica de Catalunya, 1997.

[20] V. Dalmau. Constraint satisfaction problems in non-deterministic logarithmic space. In *Proceedings of ICALP'02*, volume 2380 of *LNCS*, pages 414–425, 2002.

[21] V. Dalmau. Generalized majority-minority operations are tractable. In *Proceedings of LICS'05*, pages 438–447, 2005.

[22] V. Dalmau. A new tractable class of constraint satisfaction problems. *Annals of Mathematics and Artificial Intelligence*, 44(1-2):61–85, 2005.

[23] R. Dechter. *Constraint Processing.* Morgan Kaufmann, 2003.

[24] U. Egly, T. Eiter, H. Tompits, and S. Woltran. Solving advanced reasoning tasks using quantified Boolean formulas. In *Proceedings of AAAI'00*, pages 417–422, 2000.

[25] T. Feder and Ph.G. Kolaitis. Closures and dichotomies for quantified constraints. manuscript, 2005.

[26] T. Feder and M.Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory. *SIAM Journal on Computing*, 28:57–104, 1998.

[27] A.S. Fraenkel. Combinatorial games. *Electronic Journal of Combinatorics*, 2003. Dynamic Survey DS2.

[28] A.S. Fraenkel. Complexity, appeal and challenges of combinatorial games. *Theoretical Computer Science*, 313(3):393–415, 2004.

[29] M. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, CA., 1979.

[30] I.A. Gent, P. Nightingale, and K. Stergiou. QCSP-solve: A solver for quantified constraint satisfaction problems. In *Proceedings of IJCAI'05*, pages 138–143, 2005.

[31] E. Giunchiglia, M. Narizzano, and A. Tacchella. Backjumping for quantified Boolean logic satisfiability. *Artificial Intelligence*, 145(1-2):99–120, 2003.

[32] A. Horn. On sentences which are true of direct unions of algebras. *Journal of Symbolic Logic*, 16(1):14–21, 1951.

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

[34] P.G. Jeavons, D.A. Cohen, and M.C. Cooper. Constraints, consistency and closure. *Artificial Intelligence*, 101(1-2):251–265, 1998.

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

[36] H. Kleine Büning, M. Karpinski, and A. Flögel. Resolution for quantified Boolean formulas. *Information and Computation*, 117(1):12–18, 1995.

[37] Ph.G. Kolaitis and M.Y. Vardi. Conjunctive-query containment and constraint satisfaction. *Journal of Computer and System Sciences*, 61:302–332, 2000.

[38] Ph.G. Kolaitis and M.Y. Vardi. A game-theoretic approach to constraint satisfaction. In *Proceedings of AAAI'00*, pages 175–181, 2000.

[39] A. Krokhin, A. Bulatov, and P. Jeavons. The complexity of constraint satisfaction: an algebraic approach. In *Structural Theory of Automata, Semigroups, and Universal Algebra*, volume 207 of *NATO Science Series II: Math., Phys., Chem.*, pages 181–213. Springer Verlag, 2005.

[40] N. Mamoulis and K. Stergiou. Algorithms for quantified constraint satisfaction problems. In *Proceedings of CP'04*, volume 3258 of *LNCS*, pages 752–756, 2004.

[41] C.H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

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

[43] R. Pöschel and L.A. Kalužnin. *Funktionen- und Relationenalgebren.* DVW, Berlin, 1979.

[44] E.L. Post. *The two-valued iterative systems of mathematical logic*, volume 5 of *Annals Mathematical Studies.* Princeton University Press, 1941.

[45] O. Reingold. Undirected ST-connectivity in log-space. In *Proceedings of STOC'05*, pages 376–385, 2005.

[46] J. Rintanen. Constructing conditional plans by a theorem-prover. *Journal of Artificial Intelligence Research*, 10:323–352, 1999.

[47] M. Rychlik. On probabilistic quantified satisfiability games. In *Proceedings of MFCS'03*, volume 2747 of *LNCS*, pages 652–661, 2003.

[48] T.J. Schaefer. The complexity of satisfiability problems. In *Proceedings of STOC'78*, pages 216–226, 1978.

[49] T.J. Schaefer. On the complexity of some two-person perfect-information games. *Journal of Computer and System Sciences*, 16(2):185–225, 1978.

[50] A. Szendrei. *Clones in Universal Algebra*, volume 99 of *Seminaires de Mathematiques Superieures.* University of Montreal, 1986.

[51] R. Williams. Algorithms for quantified Boolean formulas. In *Proceedings of SODA'02*, pages 299–307, 2002.