From Pebble Games to Tractability: An Ambidextrous Consistency Algorithm for Quantified Constraint Satisfaction

Hubie Chen and Víctor Dalmau

Departament de Tecnologia
Universitat Pompeu Fabra
Barcelona, Spain
{hubie.chen,victor.dalmau}@upf.edu

Abstract. The constraint satisfaction problem (CSP) and quantified constraint satisfaction problem (QCSP) are common frameworks for the modelling of computational problems. Although they are intractable in general, a rich line of research has identified restricted cases of these problems that are tractable in polynomial time. Remarkably, many tractable cases of the CSP that have been identified are solvable by a single algorithm, which we call here the consistency algorithm. In this paper, we give a natural extension of the consistency algorithm to the QCSP setting, by making use of connections between the consistency algorithm and certain two-person pebble games. Surprisingly, we demonstrate a variety of tractability results using the algorithm, revealing unified structure among apparently different cases of the QCSP.

1 Introduction

The *constraint satisfaction problem (CSP)* is widely acknowledged as a convenient framework for modelling search problems. An instance of the CSP is a closed primitive positive formula over a relational signature, that is, a formula of the form

$$\exists v_1 \ldots \exists v_n (R(v_{i_1}, \ldots, v_{i_k}) \wedge \ldots)$$

along with a relational structure over the same signature. The question is to decide whether or not the relational structure models the formula. In the constraint satisfaction literature, the atomic formulas $R(v_{i_1}, \ldots, v_{i_k})$ in a CSP instance are called constraints, and the CSP is usually phrased as the problem of deciding whether or not there is a variable assignment satisfying all constraints in a given set. The CSP can also be defined as the problem of deciding, given an ordered pair $(\mathbf{A}, \mathbf{B})$ of relational structures, whether or not there is a relational homomorphism from $\mathbf{A}$ to $\mathbf{B}$. As the many equivalent formulations of the CSP suggest, CSPs arise naturally in a wide variety of domains, including logic, algebra, database theory, artificial intelligence, and graph coloring.

The initially given definition of the CSP leads naturally to the generalization of the CSP called the *quantified constraint satisfaction problem (QCSP)*, where both universal and existential quantification are allowed. The greater generality of the QCSP permits the modelling of a variety of computational problems that cannot be expressed using the CSP, for instance, problems from the areas of verification, non-monotonic reasoning, planning, and game playing. The higher expressiveness of the QCSP, however, comes at the price of higher complexity: the QCSP is in general complete for the complexity class PSPACE, in contrast to the CSP, which is in general complete for the complexity class NP.

The general intractability of the CSP and the QCSP has motivated a large and rich body of research aimed at identifying and understanding restricted cases of these problems that are tractable, by which we mean decidable in polynomial time. Remarkably, many tractable cases of the CSP that have been identified are solvable by parameterizations of a single algorithm which we call the *consistency algorithm*. Intuitively speaking, the consistency algorithm performs inference by continually considering bounded-size subsets of the entire set of variables, of a CSP instance.

In this paper, we give an algorithm for the QCSP that naturally extends the consistency algorithm for the CSP. Surprisingly, we are able to obtain a variety of QCSP tractability results using our algorithm, revealing unified structure among apparently different cases of the QCSP. Because of its qualitative similarity to CSP consistency as well as its ability to solve a number of QCSP generalizations of CSP instances solvable by CSP consistency methods, we believe that our algorithm can reasonably be viewed as a consistency algorithm for the QCSP, and refer to it as such.

In the rest of this section, we briefly review related work on the CSP, give a general description of our algorithm, and then describe the tractable cases of the QCSP that we demonstrate to be solvable by our algorithm.

Restrictions on the CSP. To discuss the relevant restricted cases of the CSP that have been previously studied, we view the CSP as the "relational homomorphism" problem of deciding, given an ordered pair $(\mathbf{A}, \mathbf{B})$ of relational structures, whether or not there is a homomorphism from $\mathbf{A}$ to $\mathbf{B}$. By and large, the restrictions that have been identified and studied can be placed into one of two categories, which have become known as *left-hand side* restrictions and *right-hand side* restrictions. Left-hand side restrictions, also known as *structural restrictions*, arise by considering a prespecified class $\mathcal{A}$ of relational structures from which the left-hand side structure $\mathbf{A}$ must come, whereas right-hand side restrictions, arise by considering a prespecified class $\mathcal{B}$ of relational structures from which the right-hand side structure $\mathbf{B}$ must come. When $\mathcal{B} = \{\mathbf{B}\}$ is of size one, $\mathbf{B}$ is often called the *constraint language*.

In studies of the left-hand side, the restriction of *bounded treewidth* has played a major role. Treewidth is, intuitively, a graph-theoretic measure of how "tree-like" a graph is, and the treewidth of a relational structure $\mathbf{A}$ is simply the treewidth of the *Gaifman graph* of $\mathbf{A}$, or the graph having an edge between any two elements occurring together in a tuple. The tractability of collections of relational structures $\mathcal{A}$ having *bounded treewidth* was shown by Dechter and Pearl [11] and Freuder [13]. An alternative proof was later given by Kolaitis and Vardi

[19]. Dalmau et al. [9] building on ideas of Kolaitis and Vardi [19, 20] showed that consistency is an algorithm for the CSP under bounded treewidth. They also gave a wider condition that naturally expanded the condition of bounded treewidth, and showed that the wider condition is tractable, again, via the consistency algorithm. The optimality of this latter result of Dalmau et al. was demonstrated by Grohe [16], who proved that in the case that $\mathcal{A}$ has bounded arity, if $\mathcal{A}$ gives rise to a tractable case of the CSP, then it must fall into the natural expansion identified by Dalmau et al. [9]. There has also been work on the case of unbounded arity [15].

Study of right-hand side restrictions has its origins in the paper of Schaefer [21], who classified all constraint languages $\mathbf{B}$, with a two-element universe, giving rise to a tractable case of the CSP. Much research has been directed towards classifying all constraint languages with a finite universe giving rise to a tractable CSP, and recent years have seen some major classification results [2, 3] as well as other tractability results; in many of the tractability results, such as [17, 10, 1, 4], consistency is used to demonstrate tractability.

Quantified pebble games and our new consistency algorithm. It has been demonstrated by Kolaitis and Vardi [20] that there are intimate connections between the consistency algorithm and certain combinatorial pebble games. It is from the vantage point of these pebble games that we develop our quantified analog of the consistency algorithm, and this paper overall takes strong methodological and conceptual inspiration from the paper [20] as well as from [9].

To explain some details, the consistency algorithm is a general method that adds more constraints to a CSP instance, and in some cases can detect an inconsistency (that is, that the CSP instance is not satisfiable). We have said that the consistency algorithm solves certain restricted cases of the CSP; by this, we mean that an instance is unsatisfiable if *and only if* an inconsistency is detected. In [20], consistency was linked to *existential pebble games* which are played by two players, the *Spoiler* and the *Duplicator*, on a pair of relational structures $(\mathbf{A}, \mathbf{B})$; these games were introduced by Kolaitis and Vardi [18] for analyzing the expressive power of the logic *Datalog*. The link established is that the Duplicator wins the existential pebble game if and only if no inconsistency is detected by the consistency algorithm: that is, deciding if an instance is *consistent* is equivalent to deciding if the Duplicator has a *winning strategy* for the existential pebble game.

We define a version of this pebble game which we call the *quantified pebble game* that is suited for the QCSP setting. Recall that an instance of the CSP can be viewed as a pair of relational structures $(\mathbf{A}, \mathbf{B})$; it turns out that an instance of the QCSP can be viewed as a pair of relational structures $(\mathbf{A}, \mathbf{B})$ along with a quantifier prefix $p = Q_1 a_1 \ldots Q_n a_n$ where the Q_i are quantifiers and $\{a_1, \ldots, a_n\}$ are the elements of $\mathbf{A}$. In the existential pebble game, the Spoiler places pebbles on elements of $\mathbf{A}$ and the Duplicator must respond by placing pebbles on elements of $\mathbf{B}$. A primary difference between the existential pebble game and our new pebble game is that whereas the Spoiler may, at any time, place a pebble on any element of $\mathbf{A}$ in the existential pebble game, the Spoiler must respect the quantifier prefix in placing down new pebbles in the quantified pebble game. In a sense that we will make precise, any new pebbles placed by the Spoiler must come "after" all existing pebbles in the quantifier prefix p.

After introducing our quantified pebble game and identifying some of its basic properties, we show how a consistency algorithm for the QCSP can be derived from our pebble game, analogously to the way in which the CSP consistency algorithm can be derived from the existential pebble game.

Fascinatingly and surprisingly, our QCSP consistency algorithm is ambidextrous in that it can be employed to obtain *both* left-hand and right-hand side tractability results, and hence reveals unified structure among apparently different cases of the QCSP. We now turn to describe these tractability results.

The left-hand side: bounded treewidth. As we have discussed, bounded treewidth not only gives rise to a tractable CSP, but–under the assumption of bounded arity–lies behind every left-hand side restricted CSP. However, it has recently been revealed that bounded treewidth, at least applied straightforwardly, behaves differently in the QCSP. Chen [7] recently showed that bounded treewidth also guarantees tractability in the QCSP, but under the assumption that both the universe size of $\mathbf{B}$ and the number of quantifier alternations are constant. The question of whether or not any part of this assumption could be lifted while preserving tractability was recently addressed by Gottlob et al. [14], who proved that if there is no bound on the universe size of $\mathbf{B}$, bounded treewidth instances are generally intractable. This seems to suggest a disconnect between bounded treewidth in the CSP and QCSP setting–since the tractability of bounded treewidth in the CSP setting is independent of any properties of the right-hand side structure $\mathbf{B}$–and prompts the question of whether or not there is any "pure" left-hand side restriction, generalizing bounded treewidth in the CSP, that guarantees tractability in the QCSP.

In this paper, we answer this question in the affirmative by introducing a natural generalization of bounded treewidth (in the CSP) that guarantees QCSP tractability. This generalization is easy to describe from a high-level perspective: the treewidth of a relational structure $\mathbf{A}$ can be described as the minimization of a certain quantity over *all* possible orderings of the elements of $\mathbf{A}$. Our new notion of treewidth can be described in the same way, but where the minimization is over all orderings of the $\mathbf{A}$-elements that respect the quantifier prefix (in a natural way that we make precise in the paper).

We show that instances of the QCSP having bounded treewidth in our new sense are tractable via consistency, and (in the last section of the paper) also expand this tractability result in a way that is analogous to and generalizes the expansion carried out in the CSP setting [9]. It is worth emphasizing that, as in [9], we show that our new type of bounded treewidth (and its expansion) is tractable by consistency, a method that does *not* require the computation of any form of tree decomposition.

In [14], another tractable left-hand side restricted QCSP was identified; see also [12] for closely related work.

The right-hand side. On the right-hand side, we give two tractability results via consistency. The first is the tractability of constraint languages $\mathbf{B}$ having a *near-unanimity polymorphism*; this class has been studied and shown to be tractable in the CSP [17], and is also known to be tractable in the QCSP [6]. The second concerns constraint languages $\mathbf{B}$ having a *set function polymorphism*. While such constraint languages are all tractable in the CSP setting [10], they give rise to two modes of behavior in the QCSP [8]. We show that any such constraint languages that are QCSP tractable are tractable by consistency. Both of these classes are fundamental to consistency in the CSP setting, as they both give exact characterizations of particular parameterizations of consistency that have been of interest; we refer the reader to the cited papers for more information.

2 Quantified Constraint Satisfaction

In this section, we define the quantified constraint satisfaction problem and associated notions to be used throughout the paper.

A *relational signature* is a finite set of relation symbols, each of which has an associated arity. A *relational structure* $\mathbf{A}$ (over signature σ) consists of a universe A and a relation $R^{\mathbf{A}}$ over A for each relation symbol R (of σ), such that the arity of $R^{\mathbf{A}}$ matches the arity associated to R. We refer to the elements of the universe of a relational structure $\mathbf{A}$ as $\mathbf{A}$-elements. Throughout this paper, we assume that all relational structures under discussion are finite.

A *quantified constraint formula* over signature σ is a closed first-order formula having conjunction as its only propositional connective in prenex normal form. That is, a quantified constraint formula (over σ) is a formula of the form $Q_1 v_1 \ldots Q_n v_n \psi(v_1, \ldots, v_n)$ where each Q_i is a quantifier (either $\exists$ or $\forall$) and ψ is the conjunction of expressions of the form $R(v_{i_1}, \ldots, v_{i_k})$, where R is a relation symbol from σ and k is the arity of R.

An instance of the *quantified constraint satisfaction problem (QCSP)* consists of a quantified constraint formula ϕ and a relational structure $\mathbf{B}$ over the same signature; the question is to decide whether or not $\mathbf{B} \models \phi$.

It is known that any quantified constraint formula with only existential quantifiers, known as *conjunctive queries*, can be naturally associated with a relational structure [5]. It will be conceptually and terminologically useful for us to associate an object called a *quantified relational structure* to each quantified constraint formula. A *quantified relational structure* is a pair $(p, \mathbf{A})$ where $\mathbf{A}$ is a relational structure and p is an expression of the form $Q_1 v_1 \ldots Q_n v_n$ where each Q_i is a quantifier (either $\exists$ or $\forall$) and $v_1, \ldots, v_n$ are exactly the elements of the universe of $\mathbf{A}$. The quantified relational structure $(p, \mathbf{A})$ associated to a quantified constraint formula ϕ is obtained by letting p be the quantifier prefix of ϕ and leeting $R^{\mathbf{A}}$ contain all tuples $(a_1, \ldots, a_k)$ such that $R(a_1, \ldots, a_k)$ appears as a constraint in ϕ. As an example, suppose that σ contains only a binary relation symbol E, and let ϕ be the quantified constraint formula

$$\forall v_1 \exists v_2 \forall v_3 \exists v_4 (E(v_1, v_2) \wedge E(v_2, v_3) \wedge E(v_3, v_4) \wedge E(v_4, v_1)).$$

The quantified relational structure associated to ϕ in this case is

$$(\forall v_1 \exists v_2 \forall v_3 \exists v_4, \mathbf{A})$$

where $\mathbf{A}$ has universe

$$\{v_1, v_2, v_3, v_4\}$$

and

$$E^{\mathbf{A}} = \{(v_1, v_2), (v_2, v_3), (v_3, v_4), (v_4, v_1)\}.$$

It is clearly also possible to inversely map a quantified relational structure to a quantified constraint formula, and we will speak of the quantified relational structure associated to a quantified constraint formula. Indeed, we will freely interchange between corresponding quantified constraint formulas and quantified relational structures.

We say that there is a *homomorphism* from a quantified relational structure $(p, \mathbf{A})$ to a relational structure $\mathbf{B}$ if $\mathbf{B} \models \phi$, where ϕ is the quantified constraint formula associated to $(p, \mathbf{A})$. This notion of homomorphism generalizes the usual notion of relational homomorphism. Let $\mathbf{A}$ and $\mathbf{B}$ be relational structures over the same signature; it is straightforward to verify that there is a relational homomorphism from $\mathbf{A}$ to $\mathbf{B}$ if and only if there is a homomorphism from $(p, \mathbf{A})$ to $\mathbf{B}$, where p existentially quantifies all elements of the universe of $\mathbf{A}$ (in any order). Recall that a mapping h from the universe of $\mathbf{A}$ to the universe of $\mathbf{B}$ is a relational homomorphism from $\mathbf{A}$ to $\mathbf{B}$ if for any relation symbol R and any tuple $(a_1, \ldots, a_k) \in R^{\mathbf{A}}$ of $\mathbf{A}$, it holds that $(h(a_1), \ldots, h(a_k)) \in R^{\mathbf{B}}$.

A quantifier prefix $p = Q_1 v_1 \ldots Q_n v_n$ can be viewed as the concatenation of *quantifier blocks* where quantifiers in each block are the same, and consecutive quantifier blocks have different quantifiers. For example, the quantifier prefix $\forall v_1 \forall v_2 \exists v_3 \forall v_4 \forall v_5 \exists v_6 \exists v_7 \exists v_8$, consists of four quantifier blocks: $\forall v_1 \forall v_2$, $\exists v_3$, $\forall v_4 \forall v_5$, and $\exists v_6 \exists v_7 \exists v_8$. We say that a variable v_j *comes after* a variable v_i in p if they are in the same quantifier block, or v_j is in a quantifier block following the quantifier block of v_i. Equivalently, the variable v_j comes after the variable v_i in p if one of the following conditions holds: (1) $j \geq i$, or (2) $j < i$ and all of the quantifiers $Q_j, \ldots, Q_i$ are of the same type. Notice that in a quantified constraint formula, variables in the same quantifier block can be interchanged without disrupting the semantics of the formula.

3 Pebble Games and Consistency

In previous work [20, 9] certain combinatorial two-player games called *existential pebble games* were shown to be strongly linked to and shed insight on consistency algorithms for constraint satisfaction, and were also used to understand and identify tractable cases of the CSP. Here, we introduce *quantified pebble* *games* that naturally generalize existential pebble games, and show that they can similarly be used to identify QCSP tractability results.

Our quantified pebble game is defined as follows. The game is played between two players, the *Spoiler* and the *Duplicator*, on a quantified relational structure $(p, \mathbf{A})$ and a relational structure $\mathbf{B}$, where $\mathbf{A}$ and $\mathbf{B}$ are over the same signature. Game play proceeds in rounds, and in each round one of the following occurs:

1. The Spoiler places a pebble on an existentially quantified $\mathbf{A}$-element a coming after all $\mathbf{A}$-elements that already have a pebble. In this case, the Duplicator must respond by placing a corresponding pebble, denoted by $h(a)$, on a $\mathbf{B}$-element.
2. The Spoiler places a pebble on a universally quantified $\mathbf{A}$-element a coming after all $\mathbf{A}$-elements that already have a pebble. The Spoiler then places a corresponding pebble, denoted by $h(a)$, on a $\mathbf{B}$-element.
3. The Spoiler removes a pebble from an $\mathbf{A}$-element a. In this case, the corresponding pebble $h(a)$ on $\mathbf{B}$ is removed.

When game play begins, there are no pebbles on any $\mathbf{A}$-elements, nor on any $\mathbf{B}$-elements, and so the first round is of one of the first two types. We assume that the Spoiler never places two pebbles on the same $\mathbf{A}$-element, so that h is a partial function (as opposed to a relation). The Duplicator wins the quantified pebble game if he can always ensure that h is a *projective homomorphism* from $\mathbf{A}$ to $\mathbf{B}$; otherwise, the Spoiler wins. A *projective homomorphism* (from $\mathbf{A}$ to $\mathbf{B}$) is a partial function h from the universe of $\mathbf{A}$ to the universe of $\mathbf{B}$ such that for any relation symbol R and any tuple $(a_1, \ldots, a_k) \in R^{\mathbf{A}}$ of $\mathbf{A}$, there exists a tuple $(b_1, \ldots, b^k) \in R^{\mathbf{B}}$ where $h(a_i) = b_i$ for all a_i on which h is defined.

We will be most interested in versions of the quantified pebble game where the number of pebbles that can be placed on $\mathbf{A}$-elements is bounded by a constant k, which we call the *quantified k-pebble game*. To emphasize the distinction between this bounded version and the general game, we will refer to the quantified pebble game with no restriction on the number of pebbles as the *general quantified pebble game*. Moreover, it will be useful to consider a version of the quantified pebble game called the *truth quantified pebble game*. In the truth quantified pebble game on $(Q_1 a_1 \ldots Q_n a_n, \mathbf{A})$ and $\mathbf{B}$, there are exactly n rounds; in the rth round, the Spoiler places down a pebble on the $\mathbf{A}$-element a_r, and then–depending on whether or not a_r is universally or existentially quantified–either the Spoiler or the Duplicator places down a corresponding pebble on a $\mathbf{B}$-element b_r.

Proposition 1. *The following four statements are equivalent:*

1. *There is a homomorphism from $(p, \mathbf{A})$ to $\mathbf{B}$.*
2. *The Duplicator wins the truth quantified pebble game on $(p, \mathbf{A})$ and $\mathbf{B}$.*
3. *The Duplicator wins the general quantified pebble game on $(p, \mathbf{A})$ and $\mathbf{B}$.*
4. *For all $k \geq 2$, the Duplicator wins the quantified k-pebble game on $(p, \mathbf{A})$ and $\mathbf{B}$.*

We now formalize what it means for the Duplicator to win the *quantified k-pebble game*, where at most k pebbles may be placed by the Spoiler on $\mathbf{A}$ at a time. When f is a partial function, we use $\mathsf{dom}(f)$ to denote the domain of f. A *k-projective homomorphism* is a projective homomorphism h with $|\mathsf{dom}(h)| \leq k$. Define a *winning strategy* for the Duplicator for the quantified k-pebble game on $(p, \mathbf{A})$ and $\mathbf{B}$ to be a non-empty set H of k-projective homomorphisms (from $\mathbf{A}$ to $\mathbf{B}$) having the following properties, corresponding to the three types of rounds given above:

1. For every $h \in H$ with $|\mathsf{dom}(h)| < k$ and every existentially quantified $\mathbf{A}$-element $a \notin \mathsf{dom}(h)$ coming after all elements of $\mathsf{dom}(h)$, there exists a projective homomorphism $h' \in H$ extending h with $\mathsf{dom}(h') = \mathsf{dom}(h) \cup \{a\}$.
2. For every $h \in H$ with $|\mathsf{dom}(h)| < k$, every $\mathbf{B}$-element b, and every universally quantified $\mathbf{A}$-element $a \notin \mathsf{dom}(h)$ coming after all elements of $\mathsf{dom}(h)$, there exists a projective homomorphism $h' \in H$ extending h with $\mathsf{dom}(h') = \mathsf{dom}(h) \cup \{a\}$ and $h'(a) = b$.
3. H is closed under subfunctions, that is, if $h \in H$ and h extends h', then $h' \in H$.

By taking the just-given definition of a winning strategy for the quantified k-pebble-game and excluding the requirements "$|\mathsf{dom}(h)| < k$" in items (1) and (2), we obtain the definition of a winning strategy for the general quantified pebble game.

The following fact is immediate from the just-given definition.

Proposition 2. *Let H and H' be winning strategies for the quantified k-pebble game on a quantified relational structure $(p, \mathbf{A})$ and a relational structure $\mathbf{B}$. Then, $H \cup H'$ is also a winning strategy for the quantified k-pebble game on the pair of structures.*

Proposition 2 implies that in a quantified k-pebble game, there is a *maximal winning strategy*, equal to the union of all winning strategies.

We now show how to efficiently compute winning strategies for the quantified k-pebble game.

Proposition 3. *For every fixed k, it is possible to decide in polynomial time, given a quantified relational structure $(p, \mathbf{A})$ and relational structure $\mathbf{B}$, whether or not there is a winning strategy for the Duplicator in the quantified k-pebble game.*

Proof. A straightforward way to perform the decision is as follows. Compute the set of all k-projective homomorphisms from $\mathbf{A}$ to $\mathbf{B}$, and let H denote this set. Continually eliminate from H any $h \in H$ with $|\mathsf{dom}(h)| < k$ that cannot be extended as described in parts (1) and (2) of the definition of winning strategy; and also continually eliminate any $h \in H$ whose subfunctions are not all in H. Once this procedure stabilizes, the result is either an empty set or a winning strategy. Observe that all projective homomorphisms $h_1, h_2, \ldots$ that are eliminated by this procedure cannot be a member of *any* winning strategy; the proof is a straightforward induction on i, where h_i represents the ith projective homomorphism that was eliminated. Therefore, if the result of the procedure is an empty set, there is no winning strategy. Moreover, if the result of the procedure is a winning strategy, the result is in fact the maximal winning strategy. □

We have given a polynomial-time algorithm for deciding if there is a winning strategy for the quantified k-pebble game. However, we can show that our algorithm can be used to derive even stronger consequences. The given algorithm can also be used to compute the maximal winning strategy for the general quantified pebble game. In fact, we can observe that any projective homomorphism that can be eliminated in the computation of the maximal winning strategy H_k for the quantified k-pebble game, can also be eliminated in the computation of the maximal winning strategy for the general quantified pebble game! Thus, in the general quantified pebble game, any k-projective homomorphism in a winning strategy must fall into H_k, and indeed by the "subfunction" requirement, every projective homomorphism h in a winning strategy with $|\mathsf{dom}(h)| > k$ must yield a k-projective homomorphism in H_k when restricted to a subdomain of size $\leq k$. After having computed the maximal winning strategy H_k for a QCSP instance, we can therefore add constraints of the form $R(a_1, \ldots, a_k)$ with

$$R^{\mathbf{B}} = \{(h(a_1), \ldots, h(a_k)) : h \in H_k, \mathsf{dom}(h) = \{a_1, \ldots, a_k\}\}.$$

(Technically, the way we have formalized quantified constraint satisfaction, we need to expand the signature to add these constraints.) After adding these constraints, we have not changed the truth of the QCSP instance (since we have not changed whether or not there is a winning strategy for the general game, see Proposition 1), but in general we obtain a more constrained instance. In fact, in the CSP setting, adding these extra constraints corresponds exactly to the process of establishing k-consistency [20].

The overall algorithm we have described, namely, of computing H_k and then imposing new constraints, is thus a QCSP generalization of the CSP notion of establishing k-consistency. This algorithm preserves the truth of the QCSP instance and thus can be applied in general, *to any arbitrary QCSP instance.* In the case that H_k is found to be empty, we can immediately conclude that the instance is false. In the next section, we will show that in some interesting cases, the converse holds, that is, some classes of QCSP instances have the following property: if on an instance there is a winning strategy for the Duplicator in the quantified k-pebble game, then the instance is true. When this property holds, we will say that *establishing k-consistency is a decision procedure* for the class of instances.

4 Tractability

4.1 The Left-Hand Side: Bounded Treewidth

We define a notion of treewidth for quantified relational structures. Let $(p, \mathbf{A})$ be a quantified relational structure. The *Gaifman graph* of $\mathbf{A}$ is the graph with vertex set equal to the universe A of $\mathbf{A}$ and with an edge $\{a, a'\}$ for every pair of different elements $a, a' \in A$ that occur together in a $\mathbf{A}$-tuple, by which we mean an element of $R^{\mathbf{A}}$ for some relation symbol R. A *scheme* for $(p, \mathbf{A})$ is a supergraph (A, E) of the Gaifman graph of $\mathbf{A}$ along with an ordering $a_1, \ldots, a_n$ of the elements of A such that

- the ordering $a_1, \ldots, a_n$ preserves the "after" relation, that is, if $i < j$, then a_j comes after a_i in p, and
- for any a_k, its lower numbered neighbors form a clique, that is, for all k, if $i < k$, $j < k$, $\{a_i, a_k\} \in E$, and $\{a_j, a_k\} \in E$, then $\{a_i, a_j\} \in E$.

The *width* of a scheme is the maximum, over all vertices a_k, of the size of the set $\{i : i < k, \{a_i, a_k\} \in E\}$, that is, the set containing all lower numbered neighbors of a_k. The *treewidth* of a quantified relational structure $(p, \mathbf{A})$ is the minimum width over all schemes for $(p, \mathbf{A})$.

Our definition of treewidth strictly generalizes the definition of treewidth in the study of the CSP: the treewidth of a relational structure $\mathbf{A}$, as defined (for instance) in [9], is equivalent to the treewidth of $(p, \mathbf{A})$ where p existentially quantifies all elements of $\mathbf{A}$ (in any order). Note that having only existential quantifiers in p amounts to dropping the first requirement in the definition of scheme above, since in that case, *every* ordering preserves the "after" relation.

Let $\mathsf{QCSP}[\mathsf{treewidth} < k]$ be the restriction of the QCSP problem to all instances $((p, \mathbf{A}), \mathbf{B})$ where $(p, \mathbf{A})$ has treewidth strictly less than k.

Theorem 4. *For all $k \geq 2$, establishing k-consistency is a decision procedure for $\mathsf{QCSP}[\mathsf{treewidth} < k]$.*

Proof. Let $(p, \mathbf{A}), \mathbf{B}$ be an arbitrary instance of $\mathsf{QCSP}[\mathsf{treewidth} < k]$, and suppose that there is a winning strategy H for the quantified k-pebble game. We show that there is a homomorphism from $(p, \mathbf{A})$ to $\mathbf{B}$. Let the graph (A, E) and the ordering q_s be a minimum width scheme for $(p, \mathbf{A})$, and let q be the quantifier prefix equal to q_s with the corresponding quantifiers from p placed in front of each variable. Now, $(q, \mathbf{A})$ is semantically equivalent to $(p, \mathbf{A})$, since q is equal to p up to interchanging quantifier-variable pairs *within* quantifier blocks. So it suffices to show that there is a homomorphism from $(q, \mathbf{A})$ to $\mathbf{B}$. We let n denote the size of the universe A of $\mathbf{A}$, and denote q by $Q_1 a_1 \ldots Q_n a_n$. By Proposition 1, it suffices to show that Duplicator can win the truth quantified pebble game.

The strategy for the Duplicator is to play so that in the rth round the partial function $a_i \to b_i$ determined by the pebbles, when restricted to the set containing a_r and its lower numbered neighbors (in the graph (A, E) and ordering q_s), falls into H.

We prove by induction that the Duplicator can play in this way. Assume that the Spoiler has initiated the rth round by placing a pebble on a_r. Let N_r denote all lower numbered neighbors of a_r. The partial function $a_i \to b_i$ determined by the pebbles, when restricted to N_r, gives a projective homomorphism h_r in H by the induction hypothesis. Since the scheme $((A, E), q_s)$ has width $\leq k$, the projective homomorphism h_r has domain size $|N_r| = |\mathsf{dom}(h_r)| < k$.

- Suppose that a_k is existentially quantified. By part (1) of the definition of winning strategy, there is a homomorphism h'_r extending h_r to a_k. The Duplicator can thus set b_k to be $h'_r(a_k)$.
- Suppose that a_k is universally quantified. In this case, the Spoiler sets b_k. By part (2) of the definition of winning strategy, there is a homomorphism h'_r extending h_r that maps a_k to b_k.

It remains to verify that after all n rounds have passed, the Duplicator has won the truth quantified pebble game, that is, the function h mapping a_i to b_i (the function determined by the pebbles) is a homomorphism from $\mathbf{A}$ to $\mathbf{B}$. Let $(c_1, \ldots, c_k) \in R^{\mathbf{A}}$ be a tuple of $\mathbf{A}$. Let a_r be the highest numbered element of this tuple with respect to q_s. In the rth round, it was ensured that the restriction of h to $\{a_r\} \cup N_r$ was equal to a projective homomorphism h'_r falling in H. Since $\{c_1, \ldots, c_k\} \subseteq \{a_r\} \cup N_r$, we have $(h(c_1), \ldots, h(c_k)) = (h'_r(c_1), \ldots, h'_r(c_k)) \in R^{\mathbf{B}}$. $\square$

4.2 The Right-Hand Side: Relational Restrictions

For each relational structure $\mathbf{B}$, let $\mathsf{QCSP}(\mathbf{B})$ denote the problem of deciding, given a quantified constraint formula ϕ, whether or not $\mathbf{B} \models \phi$. That is, $\mathsf{QCSP}(\mathbf{B})$ is the restricted case of the QCSP where the relational structure is fixed as $\mathbf{B}$.

Near-unanimity polymorphisms. Let $\mathbf{B}$ be a relational structure with universe B. An operation $f : B^k \to B$ is a *polymorphism* of a relational structure $\mathbf{B}$ if it is a homomorphism from $\mathbf{B}^k$ to $\mathbf{B}$.[1] A *near-unanimity operation* is an operation $f : B^k \to B$ of arity ≥ 3 satisfying the identities

$$d = f(e, d, d, \ldots, d) = f(d, e, d, \ldots, d) = \cdots.$$

These identities state that if all but (at most) one of the arguments to f are equal to d, then f outputs d.

Theorem 5. *Let $\mathbf{B}$ be a relational structure having a near-unanimity polymorphism of arity k. Establishing k-consistency is a decision procedure for $\mathsf{QCSP}(\mathbf{B})$.*

Proof. We assume that the Duplicator has a winning strategy H for the quantified k-pebble game on $(p, \mathbf{A})$, $\mathbf{B}$, and aim to show that there is a homomorphism from $(p, \mathbf{A})$ to $\mathbf{B}$. We in fact prove the following claim.

Claim. Let h be a projective homomorphism from $\mathbf{A}$ to $\mathbf{B}$ such that all restrictions of h onto a domain of size $\leq k$ is in H; and, let $a \notin \mathsf{dom}(h)$ be any variable coming after all elements of $\mathsf{dom}(h)$.

- If a is existentially quantified, then there exists a projective homomorphism h' extending h with $a \in \mathsf{dom}(h')$.

[1] Recall that when $\mathbf{A}$ and $\mathbf{B}$ are relational structures over the same vocabulary with A and B, respectively, the relational structure $\mathbf{A} \times \mathbf{B}$ is defined to have universe $A \times B$ and so that for each relation symbol R, it holds that $R^{\mathbf{A} \times \mathbf{B}} = \{((a_1, b_1), \ldots, (a_k, b_k)) : (a_1, \ldots, a_k) \in R^{\mathbf{A}}, (b_1, \ldots, b_k) \in R^{\mathbf{B}}\}$.

– If a is universally quantified, then for all $\mathbf{B}$-elements b there exists a projective homomorphism h' extending h with $h'(a) = b$.

Proving this claim suffices, since it follows immediately that the truth quantified pebble game can be won by the Duplicator, which by Proposition 1 implies that there is a homomorphism from $(p, \mathbf{A})$ to $\mathbf{B}$.

We prove the claim by induction on the size of $\mathsf{dom}(h)$. By the definition of winning strategy, the claim holds when $|\mathsf{dom}(h)| < k$. So, let us suppose that $|\mathsf{dom}(h)| \geq k$, and that $a \notin \mathsf{dom}(h)$ comes after all elements of $\mathsf{dom}(h)$. Pick any k distinct elements $a_1, \ldots, a_k \in \mathsf{dom}(h)$, and define h_i to be the restriction of h to $\mathsf{dom}(h) \setminus \{a_i\}$. We now consider two cases.

Case 1: Suppose that a is existentially quantified. By induction, each of the mappings h_i can be extended to a projective homomorphism h'_i with $a \in \mathsf{dom}(h'_i)$. Let $\mu : D^k \to D$ denote the near-unanimity polymorphism of $\mathbf{B}$. Define b to be $\mu(h'_1(a), \ldots, h'_k(a))$. We claim that the extension h' of h mapping a to b is a projective homomorphism. Let $(c_1, \ldots, c_m) \in R^{\mathbf{A}}$ be any $\mathbf{A}$-tuple. We wish to show that there is a tuple $(b_1, \ldots, b_m) \in R^{\mathbf{B}}$ such that $h'(c_j) = b_j$ for all $c_j \in \mathsf{dom}(h')$. Each h'_i is a projective homomorphism, and so for each $i = 1, \ldots, k$ there is a tuple $(b^i_1, \ldots, b^i_m) \in R^{\mathbf{B}}$ such that $h'_i(c_j) = b_j$ for all $c_j \in \mathsf{dom}(h'_i)$. We have $((b^1_1, \ldots, b^k_1), \ldots, (b^1_m, \ldots, b^k_m)) \in R^{\mathbf{B}^k}$; we claim that the desired tuple $(b_1, \ldots, b_m)$ can be obtained as $\mu((b^1_1, \ldots, b^k_1), \ldots, (b^1_m, \ldots, b^k_m))$. Let us consider any $c_j \in \mathsf{dom}(h')$.

– If $c_j = a$, then by our definition of h' we have $b_j = \mu(b^1_j, \ldots, b^k_j)$.
– If $c_j \in \{a_1, \ldots, a_k\}$ and $c_j = a_l$, then all elements $b^1_j, \ldots, b^k_j$ are equal to $h'(c_j)$ except possibly b^l_j, so $\mu(b^1_j, \ldots, b^k_j)$ is equal to $h'(c_j)$.
– If $c_j \in \mathsf{dom}(h) \setminus \{a_1, \ldots, a_k\}$, then all elements $b^1_j, \ldots, b^k_j$ are equal to $h'(c_j)$, and so $\mu(b^1_j, \ldots, b^k_j)$ is equal to $h'(c_j)$.

Case 2: Suppose that a is universally quantified and that b is a $\mathbf{B}$-element. By induction, each of the mappings h_i can be extended to a projective homomorphism h'_i with $h'_i(a) = b$. Let h' be the extension of h mapping a to b. Notice that, as in the previous case, b equals $\mu(h'_1(a), \ldots, h'_k(a))$. By reasoning as in the previous case, it can be established that h' is a projective homomorphism. □

Set function polymorphisms. When $\mathbf{B}$ is a relational structure with universe B, define $\wp(\mathbf{B})$ to be the relational structure having universe $\wp(B) \setminus \{\emptyset\}$ (where $\wp(B)$ denotes the power set of B) and such that for every relation symbol R, it holds that

$$R^{\wp(\mathbf{B})} = \{(\mathsf{pr}_1 S, \ldots, \mathsf{pr}_k S) : S \subseteq R^{\mathbf{B}}, S \neq \emptyset\}$$

where k denotes the arity of R, and $\mathsf{pr}_i S$ denotes the set $\{s_i : (s_1, \ldots, s_k) \in S\}$, that is, the projection onto the ith coordinate. We say that $f : \wp(B) \setminus \{\emptyset\} \to B$ is a *set function polymorphism* of $\mathbf{B}$ if f is a homomorphism from $\wp(\mathbf{B})$ to $\mathbf{B}$. Alternatively, one can define $f : \wp(B) \setminus \{\emptyset\} \to B$ to be a set function polymorphism of $\mathbf{B}$ if and only if all of the mappings $f_k : B^k \to B$ defined by $f_k(b_1, \ldots, b_k) = f(\{b_1, \ldots, b_k\})$ (with $k \geq 1$) are polymorphisms of $\mathbf{B}$. (It is straightforward to verify that these two definitions are equivalent.) We show that a variant of consistency called default k-consistency solves any *QCSP-tractable set function polymorphisms*, that is, a set function polymorphism giving rise to a tractable case of the QCSP [8]. Note that default k-consistency is a stronger concept than k-consistency, so establishing default k-consistency gives a unified algorithm for all tractable cases discussed in this paper. (Details are left to the full version of this paper.)

Theorem 6. *Let* $\mathbf{B}$ *be a relational structure having a QCSP-tractable set function polymorphism. Establishing default* k*-consistency is a decision procedure for* $\mathsf{QCSP}(\mathbf{B})$.

5 Homomorphisms Between Quantified Relational Structures

We now define a notion of homomorphism called *Q-homomorphism* that allows us to compare quantified relational structures. Let $(p, \mathbf{A})$ and $(p', \mathbf{A}')$ be quantified relational structures. Let X (X') denote the existentially quantified variables of $\mathbf{A}$ (respectively, $\mathbf{A}'$), and let Y (Y') denote the universally quantified variables of $\mathbf{A}$ (respectively, $\mathbf{A}'$).

A pair of functions $(f : X \to X', g : Y' \to Y)$ is a *Q-homomorphism* from $(p, \mathbf{A})$ to $(p', \mathbf{A}')$ when the following conditions hold:

- For all relation symbols R and all tuples $(a_1, \ldots, a_k) \in R^{\mathbf{A}}$, there exists a tuple $(a'_1, \ldots, a'_k) \in R^{\mathbf{A}'}$ such that, for all i, either
  - both a_i and a'_i are existentially quantified and $f(a_i) = a'_i$, or
  - both a_i and a'_i are universally quantified and $a_i = g(a'_i)$.
- The mappings f, g preserve the "after" relation in that if

$$(a_1, a'_1), (a_2, a'_2) \in \{(a, f(a)) : a \in X\} \cup \{(g(a'), a') : a' \in Y'\}$$

  and a_2 comes after a_1, then a'_2 comes after a'_1.

Q-homomorphisms and homomorphisms (from quantified relational structures to relational structures) interact well in that a natural transitivity-like property holds.

Theorem 7. *Let* $(p, \mathbf{A})$ *and* $(p', \mathbf{A}')$ *be quantified relational structures, and let* $\mathbf{B}$ *be a relational structure. If there is a Q-homomorphism from* $(p, \mathbf{A})$ *to* $(p', \mathbf{A}')$ *and there is a homomorphism from* $(p', \mathbf{A}')$ *to* $\mathbf{B}$*, then there is a homomorphism from* $(p, \mathbf{A})$ *to* $\mathbf{B}$.

Proof. (idea) Let (f, g) be a Q-homomorphism from $(p, \mathbf{A})$ to $(p', \mathbf{A}')$, and assume that the Duplicator wins the truth quantified pebble game on $(p', \mathbf{A}')$ and $\mathbf{B}$. By Proposition 1, it suffices to show that the Duplicator wins the truth quantified pebble game on $(p, \mathbf{A})$ and $\mathbf{B}$; we give a winning strategy for this game where the Duplicator simulates the truth quantified pebble game on $(p', \mathbf{A}')$ and $\mathbf{B}$.

– When the Spoiler places a pebble on an existentially quantified $\mathbf{A}$-element a, the Duplicator places a pebble on the $\mathbf{A}'$-element $f(a)$ in the simulated game, and then uses the response of the Duplicator in the simulated game to determine where on $\mathbf{B}$ to place a pebble.
– When the Spoiler places a pebble on a universally quantified $\mathbf{A}$-element a and a corresponding pebble on a $\mathbf{B}$-element b, there are two cases. If a does not occur in any $\mathbf{A}$-tuple, the Duplicator simply ignores the pebble on a, as well as the corresponding pebble on b. If a does occur in an $\mathbf{A}$-tuple, then by definition of Q-homomorphism, there exists a $\mathbf{B}$-element a' such that $a = g(a')$ and so the set $g^{-1}(a)$ is non-empty. The Duplicator then places pebbles on all elements of $g^{-1}(a)$ in the simulated game, and places down the $|g^{-1}(a)|$ corresponding pebbles all on b.

□

We say that two quantified relational structures $(p, \mathbf{A})$ and $(p', \mathbf{A}')$ are *Q-homomorphically equivalent* if there exists a Q-homomorphism from $(p, \mathbf{A})$ to $(p', \mathbf{A}')$, and there exists a Q-homomorphism from $(p', \mathbf{A}')$ to $(p, \mathbf{A})$. The following corollary is immediate from Theorem 7.

Corollary 8. *Let $(p, \mathbf{A})$ and $(p', \mathbf{A}')$ be Q-homomorphically equivalent quantified relational structures, and let $\mathbf{B}$ be a relational structure. There is a homomorphism from $(p, \mathbf{A})$ to $\mathbf{B}$ if and only if there is a homomorphism from $(p', \mathbf{A}')$ to $\mathbf{B}$.*

Theorem 7, as we have noted, generalizes Corollary 8. However, Corollary 8 can be generalized in a different direction. In particular, we can show that two Q-homomorphically equivalent quantified relational structures behave identically as regards the quantified k-pebble game.

Theorem 9. *Let $(p, \mathbf{A})$ and $(p', \mathbf{A}')$ be two Q-homomorphically equivalent quantified relational structures, and let $\mathbf{B}$ be a relational structure. For all $k \geq 2$, the Duplicator wins the quantified k-pebble game on $(p, \mathbf{A})$ and $\mathbf{B}$ if and only if the Duplicator wins the quantified k-pebble game on $(p', \mathbf{A}')$ and $\mathbf{B}$.*

Proof. (idea) Assume that the Duplicator wins the quantified k-pebble game on $(p', \mathbf{A}')$ and $\mathbf{B}$. We want to show that the Duplicator wins the quantified k-pebble game on $(p, \mathbf{A})$ and $\mathbf{B}$. We may assume without loss of generality that all universally quantified variables in $(p', \mathbf{A}')$ appear in a tuple of $\mathbf{A}'$, and likewise for $(p, \mathbf{A})$. Let (f, g) be a Q-homomorphism from $(p, \mathbf{A})$ to $(p', \mathbf{A}')$. As $(p, \mathbf{A})$ and $(p', \mathbf{A}')$ are Q-homomorphically equivalent, it can be verified that g is a bijection between the universally quantified variables of $(p', \mathbf{A}')$ and those of $(p, \mathbf{A})$. We can show that the Duplicator wins the quantified k-pebble game on $(p, \mathbf{A})$ and $\mathbf{B}$ using the same simulation strategy as in the proof of Theorem 7. Note that because g is a bijection, no more than k pebbles are ever placed on $\mathbf{A}'$ in the simulated game (which is played on $(p', \mathbf{A}')$ and $\mathbf{B}$). □

We remark that Corollary 8 can be derived from Theorem 9 (along with Proposition 1).

The results obtained in this section allow us to expand the tractability result concerning instances with bounded treewidth (Theorem 4). This result generalizes an analogous expansion that was carried out in the CSP setting [9]. Let $\mathsf{QCSP}[\mathcal{H}(\mathsf{treewidth} < k)]$ be the restriction of the QCSP problem to all instances $((p, \mathbf{A}), \mathbf{B})$ where $(p, \mathbf{A})$ is Q-homomorphically equivalent to a quantified relational structure with treewidth strictly less than k.

Theorem 10. *For all $k \geq 2$, establishing k-consistency is a decision procedure for $\mathsf{QCSP}[\mathcal{H}(\mathsf{treewidth} < k)]$.*

Proof. Immediate from Theorem 4, Corollary 8, and Theorem 9. □

References

1. Andrei Bulatov. Combinatorial problems raised from 2-semilattices. Manuscript.
2. Andrei Bulatov. A dichotomy theorem for constraints on a three-element set. In *Proceedings of 43rd IEEE Symposium on Foundations of Computer Science*, pages 649–658, 2002.
3. Andrei Bulatov. Tractable conservative constraint satisfaction problems. In *Proceedings of 18th IEEE Symposium on Logic in Computer Science (LICS '03)*, pages 321–330, 2003. Extended version appears as Oxford University technical report PRG-RR–03-01.
4. Andrei Bulatov. A graph of a relational structure and constraint satisfaction problems. In *Proceedings of 19th IEEE Annual Symposium on Logic in Computer Science (LICS'04)*, 2004.
5. Ashok Chandra and Philip Merlin. Optimal implementation of conjunctive queries in relational data bases. In *STOC*, 1977.
6. Hubie Chen. *The Computational Complexity of Quantified Constraint Satisfaction*. PhD thesis, Cornell University, August 2004.
7. Hubie Chen. Quantified constraint satisfaction and bounded treewidth. In *ECAI*, 2004.
8. Hubie Chen. Quantified constraint satisfaction, maximal constraint languages, and symmetric polymorphisms. In *STACS*, 2005.
9. Victor Dalmau, Phokion G. Kolaitis, and Moshe Y. Vardi. Constraint satisfaction, bounded treewidth, and finite-variable logics. In *Constraint Programming '02*, LNCS, 2002.
10. Victor Dalmau and Justin Pearson. Closure functions and width 1 problems. In *CP 1999*, pages 159–173, 1999.
11. Rina Dechter and Judea Pearl. Tree clustering for constraint networks. *Artificial Intelligence*, pages 353–366, 1989.
12. J. Flum, M. Frick, and M. Grohe. Query evaluation via tree-decompositions. *JACM*, 2002.
13. Eugene Freuder. Complexity of k-tree structured constraint satisfaction problems. In *AAAI-90*, 1990.
14. Georg Gottlob, Gianluigi Greco, and Francesco Scarcello. The complexity of quantified constraint satisfaction problems under structural restrictions. In *IJCAI*, 2005.
15. Georg Gottlob, Nicola Leone, and Francesco Scarcello. A comparison of structural csp decomposition methods. *Artif. Intell.*, 124(2):243–282, 2000.

16. Martin Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. In *FOCS 2003*, pages 552–561, 2003.
17. Peter Jeavons, David Cohen, and Martin Cooper. Constraints, consistency, and closure. *Articial Intelligence*, 101(1-2):251–265, 1998.
18. Ph.G. Kolaitis and M.Y. Vardi. On the expressive power of Datalog: tools and a case study. *Journal of Computer and System Sciences*, 51(1):110–134, 1995.
19. Ph.G. Kolaitis and M.Y. Vardi. Conjunctive-query containment and constraint satisfaction. *Journal of Computer and System Sciences*, 61:302–332, 2000.
20. Ph.G. Kolaitis and M.Y. Vardi. A game-theoretic approach to constraint satisfaction. In *Proceedings 17th National (US) Conference on Artificial Intellignece, AAAI'00*, pages 175–181, 2000.
21. Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 216–226, 1978.