Quantified Equality Constraints[*]

Manuel Bodirsky
Laboratoire d'informatique (LIX, CNRS UMR 6171),
École Polytechnique, Paris-Palaiseau, France
bodirsky@lix.polytechnique.fr

Hubie Chen
Departament de Tecnologies de la Informació i les Comunicacions
Universitat Pompeu Fabra, Barcelona, Spain
hubie.chen@upf.edu

September 3, 2010

Abstract

An *equality template* is a relational structure with infinite universe whose relations can be defined by boolean combinations of equalities. We prove a complexity classification for *quantified constraint satisfaction problems (QCSPs)* over equality templates: these problems are in L (decidable in logarithmic space), NP-hard, or coNP-hard. To establish our classification theorem we combine methods from universal algebra with concepts from model theory.

1 Introduction

The *constraint satisfaction problem (CSP)* is the problem of deciding the truth of a primitive positive (pp) sentence

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

over a finite relational signature, relative to a relational structure over the same signature. For a relational structure Γ, define CSP(Γ) to be the CSP where the relational structure is fixed to be Γ. Note that the instances of CSP(Γ) are syntactic objects, namely, the pp-sentences over the signature of Γ. When discussing problems of the form CSP(Γ), we call Γ the *template*. In a now oft-cited result, Schaefer [25] studied the family of problems CSP(Γ) over templates with a two-element universe. He proved a *dichotomy theorem*: every such problem CSP(Γ) is either in P, or is NP-complete. In recent years, there has been considerable research effort directed towards classifying the complexity of CSP(Γ) over all finite templates Γ [1, 7–10, 13, 15, 16, 19, 23].

One of Schaefer's motivations was to provide a tool for developing NP-hardness proofs; in particular, one can show that a problem is hard by showing that it can simulate conjunctions of atomic formulas over any template Γ such that CSP(Γ) is known to be NP-hard. Another feature of his and subsequent work is that it places tractable cases of CSP(Γ) into a unified framework; the tractable cases of CSP(Γ) encompassed by Schaefer's theorem include 2-SAT and HORN SAT, two classic tractable fragments of propositional logic.

In recent work, Bodirsky and Kára [4] gave a systematic CSP(Γ) classification result for templates Γ over an *infinite* universe. They studied the fundamental class of *equality* templates, defined to be templates where every relation is definable from equalities ($x = y$) and the usual boolean connectives. They proved a complete complexity classification on equality templates, showing that each problem CSP(Γ) is either in P or NP-complete.

[*]An extended abstract of this paper appeared in the Proceedings of LICS 2007

In this paper, we consider the *quantified constraint satisfaction problem (QCSP)*, the natural generalization of the CSP where universal quantification is permitted in addition to existential quantification. We establish a complexity classification for the problems QCSP(Γ) over equality templates, showing that each such problem is either in L (decidable in logarithmic space), NP-complete, or coNP-hard. We believe that this classification result can, in analogy to Schaefer's theorem, serve as a tool for performing complexity analysis on logical formulas involving both quantifiers. The formulas that we study can be viewed as syntactically restricted fragments of the first-order theory of equality. In fact, for the templates Γ that we show to be NP-complete, containment in NP follows from a result of Kozen [20] showing that *positive* first-order sentences have an NP-complete validity problem. To the best of our knowledge, our algorithmic result showing certain formulas to be decidable in logarithmic space, is new.

The classification. We now describe the classes of formulas corresponding to the three complexity regimes of our classification. First, we say that a relation is *negative* if it can be defined by a quantifier-free equality formula that is the conjunction of equalities and disjunctions of disequalities $x_1 \neq y_1 \vee \cdots \vee x_k \neq y_k$. We extend this to templates by saying that a template is negative if all of its relations are negative. We show that, for all negative templates Γ, the problem QCSP(Γ) is in L.

Let D be (here and throughout) a set of arbitrary infinite cardinality.

Example 1.1 *Let Γ_1 be the relational structure $(D; T, =)$, where T is the ternary relation*

$$T = \{(x, y, z) \in D^3 \mid (x \neq y \vee y \neq z)\}.$$

The relation T is negative, as it may be viewed as the conjunction of a single disjunction of disequalities. An example of an instance of QCSP(Γ_1) is

$$\forall y_1 \exists x_2 \forall y_3 \exists x_4 \forall y_5 \exists x_6$$
$$(T(x_2, y_5, x_4) \ \wedge \ x_4 = y_1 \ \wedge \ T(y_1, x_6, y_3)) \ .$$

It can be verified that the formula above is true in Γ_1. Since Γ_1 is negative, it follows from our classification theorem that QCSP(Γ_1) is in L.

The second type of relations we identify are *positive* relations. We say that a relation is *positive* if it can be defined by a quantifier-free equality formula that uses only the positive connectives AND ($\wedge$) and OR ($\vee$). As before, we say that a template is positive if all of its relations are positive. We show that, for all templates Γ that are positive but not negative, the problem QCSP(Γ) is NP-complete.

Example 1.2 *Let Γ_2 be the relational structure $(D; P)$, where P is the ternary relation*

$$P = \{(x, y, z) \in D^3 \mid (x = y) \vee (y = z)\}.$$

The relation P, and hence the constraint language Γ_2, is positive. Note that Γ_2 is not negative. First, it is clear that P does not imply that some of its arguments are equal, and therefore we only have to show that it cannot be defined by a disjunction of disequalities. But any relation defined by a disjunction of disequalities contains the tuple where all entries are distinct, and the relation P does not contain such a tuple. Thus, our classification theorem implies that QCSP(Γ_2) is NP-complete.

Finally, we show that the remaining templates Γ–those not falling into either of the two previously identified classes–give rise to a problem QCSP(Γ) that is coNP-hard.

Example 1.3 *Let Γ_3 be the relational structure $(D; S)$, where S is the ternary relation*

$$S = \{(x, y, z) \in D^3 \mid (x = y \wedge y = z)$$
$$\vee \ (x \neq y \ \wedge \ y \neq z \ \wedge \ x \neq z)\}.$$

It can be verified that S is not positive. Clearly, every positive relation contains all tuples where all entries are equal, but S does not contain such tuples. It also does not contain the all-different tuple, and is not negative. Hence, by our classification theorem, the problem QCSP(Γ_3) is coNP-hard.

Techniques used. As we mentioned, the complexity of the CSP over equality templates has been classified recently [4]. This result was obtained by a universal-algebraic approach originally developed for studying CSPs over finite domains [7–10, 13, 16]. The approach rests on studying certain closure properties, so-called *polymorphisms*, of a template. The set of all polymorphisms of a template forms an algebraic object called a *clone*. For templates over infinite domains, this clone is *locally closed*. The result in [4] exhibits two polymorphisms for equality templates that imply polynomial-time decidability of the corresponding constraint satisfaction problems, and proves that the constraint satisfaction problem is NP-complete in all other cases.

It is worth noting that the polymorphisms that guarantee polynomial-time tractability for the CSP do not guarantee tractability in the QCSP. To derive our complexity classification for the QCSP, we have to obtain deeper knowledge of the associated clones. In doing so, we develop a number of new techniques; several of them could be of independent interest in universal algebra. Indeed, our results reveal new information on the lattice of clones containing all permutations, which includes all clones corresponding to equality templates. Interestingly, some of our results utilize a mix of "relational" arguments, arguments directly on the relations of a template, and "operational" arguments, arguments on the polymorphisms of a template; see Section 8 as an example.

In our proof we use a *surjective preservation theorem* for equality templates, which concerns the expressibility of a relation from formulas that may use conjunction and arbitrary (both universal and existential) quantification. In particular, this theorem shows that a relation can be defined by a first-order formula of the described form over an equality template Γ if and only if it is preserved by all *surjective* polymorphisms of Γ. An analogous theorem for *finite* templates Γ has been shown previously [6]. The proof that this result holds for all *ω-categorical* templates in the conference version is flawed (see also Section 4), and we have to leave this as an open question.

2 Preliminaries

In this section we recall fundamental notions from logic and universal algebra; the terminology and notation we use is standard and can be found for instance in [14, 22] and [27].

Relational structures. A *relational signature* τ is a (in this paper always finite) set of *relation symbols* R_i, each of which has an associated finite *arity* k_i. A *relational structure* Γ over the signature τ (also called *τ-structure*) is a set D_Γ (the *domain*) together with a relation $R_i \subseteq D_\Gamma^{k_i}$ for each relation symbol R_i from τ. For simplicity, we use the same symbol for a relation symbol and the corresponding relation. If necessary, we write R^Γ to indicate that we are talking about the relation R belonging to the structure Γ.

Automorphisms and Homomorphisms. Let Γ and Γ' be τ-structures. Isomorphisms from Γ to Γ are called *automorphisms*. The set of all automorphisms of a structure Γ forms a group with respect to composition of automorphisms. An *orbit* of a k-tuple t in Γ is the set of all tuples of the form $(\pi(t_1), \dots, \pi(t_k))$, where π is an automorphism of Γ.

A *homomorphism* from Γ to Γ' is a function f from D_Γ to $D_{\Gamma'}$ such that for each n-ary relation symbol R in τ and each n-tuple $(a_1, \dots, a_n)$, if $(a_1, \dots, a_n) \in R^\Gamma$, then $(f(a_1), \dots, f(a_n)) \in R^{\Gamma'}$. In this case we say that the map f *preserves* the relation R (and otherwise, that f *violates* R).

First-order logic. Let τ be a signature. A *τ-formula* is a first-order formula whose atomic formulas are either of the form $x = y$ or $R(x_1, \dots, x_k)$ for $R \in \tau$. A τ-sentence is a τ-formula without free variables. A first-order theory (over the signature τ) is a set of first-order τ-sentences.

If Γ is a τ-structure, then $\mathrm{Th}(\Gamma)$ denotes the *first-order theory of* Γ, that is, the set of all first-order τ-sentences that are true in Γ. A τ-structure Γ is a *model* of a theory T is all sentences in T are true in Γ.

Categoricity. For a cardinal κ, a first-order theory theory T is called *κ-categorical* if all models of T (i.e., all structures that satisfy all sentences in T) of cardinality κ are isomorphic, and T is called *totally categorical* if T is κ-categorical for all infinite cardinals κ. If Γ is a structure of cardinality κ, and if the first-order theory $\mathrm{Th}(\Gamma)$ is κ-categorical, we also say that Γ is κ-categorical.

Polymorphisms. Let D be a set, and let O be the set of *finitary operations* on D, i.e., functions from D^k to D for finite k. We say that a k-ary operation $f \in O$ *preserves* an m-ary relation $R \subseteq D^m$ if whenever $R(x^i_1, \ldots, x^i_m)$ holds for all $1 \leq i \leq k$ in Γ, then $R\big(f(x^1_1, \ldots, x^k_1), \ldots, f(x^1_m, \ldots, x^k_m)\big)$ holds in Γ. If f preserves all relations of a relational τ-structure Γ, we say that f is a polymorphism of Γ. Equivalently, f is a polymorphism of Γ if it is a homomorphism from $\Gamma^k = \Gamma \times \cdots \times \Gamma$ to Γ, where $\Gamma_1 \times \Gamma_2$ is the *(direct) product* of the two relational τ-structures Γ_1 and Γ_2 [14]; this product is also called the *categorical product* or the *cross product* [13]. Hence, the unary bijective polymorphisms are the automorphisms of Γ. We use $\mathsf{Pol}(\Gamma)$ to denote the polymorphisms of Γ, and $\mathsf{sPol}(\Gamma)$ to denote the surjective polymorphisms of Γ.

Clones. An operation π is a *projection* if for all n-tuples, $\pi(x_1, \ldots, x_n) = x_i$ for some fixed $i \in \{1, \ldots, n\}$. The *composition* of a k-ary operation f and k operations $g_1, \ldots, g_k$ of arity n is an n-ary operation defined by

$$f(g_1, \ldots, g_k)(x_1, \ldots, x_n) = \\ f\big(g_1(x_1, \ldots, x_n), \ldots, g_k(x_1, \ldots, x_n)\big).$$

A *clone* F is a set of operations defined on a set D (the *domain* of F) that is closed under compositions and that contains all projections. It is easy to verify that the set $\mathsf{Pol}(\Gamma)$ of all polymorphisms of Γ is a clone with domain D_Γ. Moreover, $\mathsf{Pol}(\Gamma)$ is also *locally closed* (see e.g. Proposition 1.6 in [27]), a property that is defined as follows. We say that an operation f is *interpolated* by a set of operations F over D if for every finite subset B of D there is some operation $g \in F$ such that $f(t) = g(t)$ for every $t \in B^k$. The set of operations that are interpolated by F is called the *local closure* of F; if F equals its local closure, we say that F is *locally closed*.

In this paper we focus on clones on a countably infinite domain D that contain all permutations of D. When clear from context, we slightly abuse terminology and say that an operation g on a countable infinite domain D is *generated* by a set of operations F (by an operation f) on D if g is contained in the smallest locally closed clone containing F (containing f) and containing all permutations of D.

Lemma 2.1 *Every operation f with infinite range generates a surjective operation g that generates f.*

Proof. Let i be a bijection between the range of f and D. We claim that the surjective operation $g := i(f)$ is generated by f. For each finite subset S of the domain, the restriction of i to S can be extended to a permutation α of D, so $\alpha(f(x)) = g(x)$ for all $x \in S$, and by local closure we have shown the claim. Similarly one can show that the operation g also generates f. $\Box$

An operation f is *essentially unary* if there is a unary operation f_0 such that $f(x_1, \ldots, x_k) = f_0(x_i)$ for some fixed $i \in \{1, \ldots, k\}$. We say that a k-ary operation f *depends on argument* i if there is no $k-1$-ary operation f' such that $f(x_1, \ldots, x_k) = f'(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k)$. We can equivalently characterize k-ary operations that depend on the i-th argument by requiring that there are elements $x_1, \ldots, x_k$ and x'_i such that $f(x_1, \ldots, x_k) \neq f(x_1, \ldots, x_{i-1}, x'_i, x_{i+1}, \ldots, x_k)$. Hence, an essentially unary operation is an operation that depends on one argument only. More generally, an operation that depends on at least k arguments is called *essentially k-ary*. We will call an essentially k-ary operation having $k \geq 2$ an *essential operation*, as is common in universal algebra. Note that every operation is either essential or essentially unary.

3 Quantified Constraint Satisfaction

A *template* or *constraint language* is simply a relational structure; we typically refer to a relational structure Γ as a *template* or *constraint language* when we are interested in the computational problem QCSP(Γ), which is defined below. (We use the terms "template" and "constraint language" synonymously.) Recall that signatures τ in this paper are assumed to contain finitely many symbols.

A first-order τ-formula is a *quantified constraint formula* if it has the form

$$Q_1 v_1 \dots Q_n v_n (\psi_1 \wedge \dots \wedge \psi_m),$$

where each Q_i is a quantifier from $\{\forall, \exists\}$, and each ψ_i is an atomic τ-formula (a formula $x = y$ or $R(x_1, \dots, x_k)$ for $R \in \tau$) that can contain both free variables and quantified variables from $\{v_1, \dots, v_n\}$.

The *quantified constraint satisfaction problem* for Γ, denoted by QCSP(Γ), is the problem of deciding, given a quantified constraint τ-formula without free variables, whether or not the formula is true in Γ. Note that both the universal and existential quantification is understood to take place over the entire universe of Γ.

Let R be a k-ary relation and let Γ be a τ-structure. We say that R has a $\forall\exists\wedge$-*definition* (or is $\forall\exists\wedge$-*definable*[1].) in Γ if there exists a quantified constraint formula ϕ with free variables $x_1, \dots, x_k$ such that $R(x_1, \dots, x_k) = \phi(x_1, \dots, x_k)$. We say that R has a pp-definition (or is pp-definable) in Γ if there exists a quantified constraint formula ϕ that uses only existential quantification with free variables $x_1, \dots, x_k$ such that $R(x_1, \dots, x_k) = \phi(x_1, \dots, x_k)$. Here, "pp" stands for "primitive positive". We use $[\Gamma]$ to denote the expansion of Γ by all relations that are $\forall\exists\wedge$-definable from Γ.

In the following, we frequently make use of the following result, which was shown in [6] for constraint languages over a finite domain, but which is purely syntactic and holds for infinite domain constraint languages as well.

Lemma 3.1 *Let Γ_1, Γ_2 be constraint languages. If every relation in Γ_1 has a $\forall\exists\wedge$-definition in Γ_2 (that is, $\Gamma_1 \subseteq [\Gamma_2]$), then QCSP($\Gamma_1$) is logarithmic space reducible to QCSP(Γ_2).*

In this paper we focus on the QCSP of a fundamental class of highly symmetric ω-categorical constraint languages, called *equality constraint languages*. An *equality-definable relation* is a relation (on an infinite domain) that has a first-order definition by an *equality formula*, i.e., a first-order formula where all atomic subformulas are of the form $x = y$. An *equality constraint language* is a relational structure on a countably infinite universe such that all of its relations are equality-definable relations.

In general, we use the symbol D to denote an infinite domain. For every constraint language Γ over an infinite domain there exists a constraint language Γ' over a *countably infinite* domain such that Γ and Γ' have the same QCSP; this follows from downward Löwenheim-Skolem theorem (see for example [14]). For equality constraint languages we even have that the complexity classification is the same for all infinite domains D. This can be seen as follows. Let D and D' be two infinite sets (not necessarily of the same cardinality). Suppose that $\phi_1, \dots, \phi_s$ are equality formulas. Let $R_1, \dots, R_s$ be the relations defined by $\phi_1, \dots, \phi_s$ over D, and let $R'_1, \dots, R'_s$ be the relations defined by $\phi_1, \dots, \phi_s$ over D'. Then QCSP($(D; R_1, \dots, R_s)$) and QCSP($(D'; R'_1, \dots, R'_s)$) are the same computational problem: an instance Φ of QCSP($(D; R_1, \dots, R_s)$) is true if and only if the first-order sentence obtained from Φ by replacing the symbols R_i by the formula ϕ_i is true in $(D; =)$, which in turn is the case if and only if this sentence is true in $(D'; =)$, and this is true if and only if Φ is true in $(D'; R'_1, \dots, R'_s)$. So, for the main result it is sufficient to prove the classification only for countable equality constraint languages, and to deduce the general case from the countable case.

The structure $(D; =)$ admits quantifier elimination [14]: this is, all equality formulas are logically equivalent to a quantifier-free equality formula. Therefore, equality constraint languages can

[1]In the conference version of the paper, motivated by latex symbol names, we wrote *few-definition* instead of $\forall\exists\wedge$-definition

be introduced equivalently as those relational structures where all relations can be defined by a boolean combination of atoms of the form $x = y$. It is also straightforward to verify that the first-order theory of equality constraint languages is totally categorical.

When Γ is an equality constraint language with domain D, any permutation of D is an automorphism of Γ, that is, the automorphism group of Γ is the full symmetric group on D. It is easy to see that all injective unary operations on D are polymorphisms of an equality constraint language Γ on D. In fact, a relational structure is an equality constraint language if and only if it is preserved by all injective unary operations on D.

The quantified constraint satisfaction problem for equality constraint languages over an infinite domain D is in PSPACE. This is closely related to the observation by Stockmeyer and Meyer that the first-order logic of equality has a validity problem that is contained in PSPACE [26]. For completeness, we sketch a proof that the problems under study are in PSPACE.

Proposition 3.2 *For every equality constraint language Γ, the problem QCSP(Γ) is in PSPACE.*

Proof. To solve the problem QCSP(Γ) in polynomial space, we can use a standard backtracking algorithm which always branches on the outermost quantifier. The key point is that, when the algorithm is about to branch on a variable v, the previous variables form an equivalence relation (where two variables are equal simply if they were assigned the same value), and as far as satisfying equality constraints is concerned, all that matters is whether or not v is set equal to the value of one of the equivalence classes, or set equal to a new value. Clearly, at each branching point there are at most linearly many choices, and so the backtrack search may be performed in polynomial space. $\Box$

4 The Surjective Preservation Theorem

It was proved in [6] that a relation R has a $\forall\exists\wedge$-definition in a *finite* structure Γ if and only if R is preserved by all surjective polymorphisms of Γ. It turns out that the same characterization holds for equality templates Γ (Theorem 4.3). In the conference version of this paper we suggested to derive this from a corresponding model-theoretic preservation theorem by [18]. But the proof contained a mistake: the model-theoretic preservation theorem involves taking *infinitary* powers of structures, whereas we want to obtain a preservation theorem for *finitary* polymorphisms.

The proof of the surjective preservation theorem for equality templates rests on the following, which holds more generally for all ω-categorical structures, not just for equality templates.

Lemma 4.1 *Let Γ be an ω-categorical structure. Suppose there exists a k such that every polymorphism of Γ of range larger than k is generated by surjective polymorphisms of Γ. Then any relation that is preserved by the surjective polymorphisms of Γ has a $\forall\exists\wedge$-definition in Γ.*

Proof. Let R be an m-ary relation preserved by the surjective polymorphisms of Γ. Consider the relation $R' = R \times D^{m+1}$. Let S be the smallest pp-definable relation containing R'. Since Γ is ω-categorical, S can be characterized as the set of all tuples $f(t_1, \dots, t_n)$ where $t_1, \dots, t_n \in R'$ and f is a polymorphism of Γ (see [5]).

We claim that $R(x_1, \dots, x_m)$ is equivalent to $\forall y_1 \dots y_{m+1}.S(x_1, \dots, x_m, y_1, \dots, y_{m+1})$. Obviously, every tuple in R satisfies this formula. For the converse, suppose that t satisfies the formula. Then the tuple $(t, a_1, \dots, a_{m+1})$ where the a_i are pairwise distinct and also different from any value in t, is contained in S. The tuple $(t, a_1, \dots, a_{m+1})$ is equal to $f(t_1, \dots, t_n)$ as described above. Then f must have range larger than m, and by assumption it follows that f is generated by surjective polymorphisms of Γ. Since R is preserved by the surjective polymorphisms of Γ, it follows that $t \in R$. $\Box$

To derive the surjective preservation theorem from the previous result, we need a fact from joint work of the authors with Michael Pinsker [3], which itself uses a theorem of Haddad and Rosenberg [12] about clones on *finite* domains containing all permutations; Haddad and Rosenberg credit the proof ideas to Malt'sev [21].

Theorem 4.2 (From [3], Lemma 36) *Let f be an essential operation on D of range k with $3 \leq k < \omega$. Then f generates all operations which take at most k values.*

We can now prove the surjective preservation theorem for equality templates.

Theorem 4.3 *A relation R has a $\forall\exists\wedge$-definition over an equality template Γ if and only if R is preserved by all surjective polymorphisms of Γ.*

Proof. First, observe that every relation R with a $\forall\exists\wedge$-definition in an arbitrary Γ is preserved by all surjective polymorphisms of Γ. This can be shown in a straightforward way by induction on the structure of the $\forall\exists\wedge$-definition of R.

For the converse, suppose that R is preserved by all surjective polymorphisms of Γ. By Lemma 4.1, it suffices to show that there exists a k such that all polymorphisms of Γ of range greater than k are generated by surjective polymorphisms. Suppose for contradiction that for every k there exists a polymorphism f_k of range larger than k that is not generated by a surjective polymorphism of Γ. By Lemma 2.1, f_k must have finite range. We distinguish two cases.

- For infinitely many k, the operation f_k is essentially unary. These essentially unary f_k cannot be injective with respect to the corresponding unary operations, for then these f_k would be generated by permutations. It follows by local closure that these f_k generate a unary surjective non-injective operation, which in turn generates all unary operations; this contradicts our assumption.
- For infinitely many k, the operation f_k is essential. Hence, Γ has for arbitrarily large k an essential polymorphism of finite range larger than k, which by Theorem 4.2 generates all operations of range k. By local closure, Γ is preserved by all operations. Since for every operation there is a surjective operation that generates it, Γ therefore has surjective polymorphisms that generate f_k (for all k), a contradiction to the choice of f_k.

□

Corollary 4.4 *Let Γ_1, Γ_2 be equality constraint languages over the same infinite domain D. If $\mathsf{sPol}(\Gamma_2) \subseteq \mathsf{sPol}(\Gamma_1)$, then QCSP$(\Gamma_1)$ is logarithmic space reducible to QCSP(Γ_2).*

Proof. Directly from Lemma 3.1 and Theorem 4.3. □

5 Classification Theorem

This section presents the statement of our classification theorem for equality constraint languages. We first identify two subclasses of equality constraint languages that will be needed to state the theorem.

Definition 5.1 *An equality-definable relation R is called* negative *if it is definable by a quantifier-free conjunction of equalities and disjunctions of disequalities. An equality constraint language Γ is* negative *if every one of its relations is negative.*

Example 5.2 *Let $S \subseteq D^7$ be the relation defined by $S(x_1, x_2, x_3, x_4, x_5, x_6, x_7) \equiv (x_1 \neq x_2 \vee x_2 \neq x_3) \wedge (x_5 \neq x_4 \vee x_4 \neq x_3) \wedge (x_1 = x_6) \wedge (x_3 = x_7)$. From the form of the definition, we can see that S is negative.*

The second subclass is the class of all *positive* equality-definable relations.

Definition 5.3 *An equality-definable relation R is* positive *if it is definable by a formula consisting of equalities and the connectives $\wedge$ and $\vee$. An equality constraint language Γ is* positive *if every one of its relations is positive.*

Example 5.4 *Let $S \subseteq D^4$ be the relation defined by $S(x_1, x_2, x_3, x_4) \equiv (x_1 = x_2) \vee (x_2 = x_3 \wedge x_3 = x_4)$. The relation S is positive.*

The following is the statement of our classification theorem.

Theorem 5.5 *(Classification theorem) Let Γ be an equality constraint language.*

- *If Γ is negative, then the problem QCSP(Γ) is decidable in logarithmic space.*
- *Else (i.e., if Γ is not negative), and if Γ is positive, then the problem QCSP(Γ) is NP-complete.*
- *Else (i.e., if Γ is neither negative nor positive), the problem QCSP(Γ) is coNP-hard.*

In the second and third cases, the problem QCSP(Γ) is complete for the respective complexity class under logarithmic space reducibility.

We now give a proof of the classification theorem which contains forward references to the needed results presented in the subsequent sections. This proof may well serve as a road map for the next four sections of the paper, which together establish the classification theorem. For ease of readability, in each of these sections, the main result of the section is stated as the first theorem. Our proof of the classification theorem only refers to these "first theorems".

Proof. If the constraint language Γ is negative, then Theorem 6.1 of Section 6 shows that QCSP(Γ) is decidable in logarithmic space. If the constraint language Γ is not negative but is positive, then Theorem 7.1 shows that QCSP(Γ) is NP-complete. Otherwise (that is, if the constraint language is neither negative nor positive), Theorem 8.1 shows that $[\Gamma]$ contains the relation $I \subseteq D^3$ (defined in Section 8). By Theorem 9.1, we have that QCSP$((D; I))$ is coNP-hard; QCSP$((D; I))$ reduces to QCSP(Γ) by Lemma 3.1. $\square$

6 Negative Languages

In this section, we present an algorithmic result for negative constraint languages.

Theorem 6.1 *If Γ is a negative constraint language, then QCSP(Γ) is decidable in logarithmic space.*

Let Γ be negative, and let Φ be an instance of QCSP(Γ) with variables $X = \{x_1, \ldots, x_n\}$. If $R(v_1, \ldots, v_k)$ is a constraint in Φ, we use $\phi_{R(v_1,\ldots,v_k)}$ to denote the formula with variables $v_1, \ldots, v_k$ that defines R as a conjunction of equalities and disjunctions of disequalities. We say that a disjunction of disequalities ψ *appears* in Φ if it is a member of one of the conjunctions $\phi_{R(\overline{v})}$ with $R(\overline{v})$ a constraint of Φ. We will make use of the following graphs.

- The undirected graph $G_\Phi^=$ has vertex set X, and for each pair $\{x, y\} \subseteq X$, the edge $\{x, y\}$ is present if and only if there exists a constraint $R(\overline{v})$ in Φ such that $\phi_{R(\overline{v})}$ contains $x = y$.
- Let ψ be a disjunction of disequalities appearing in Φ. The undirected graph G_Φ^ψ is defined to have vertex set X, and contains all edges of $G_\Phi^=$ as edges; in addition, the edge $\{x, y\}$ is present if the disequality $x \neq y$ is part of the disjunction ψ.

Our logarithmic space decision procedure is based on the following characterization of true instances. This characterization is given by "forbidden paths": an instance is true if and only if the graphs $G_\Phi^=$ and G_Φ^ψ omit paths of certain types.

For the correctness proof of the following algorithm, but also for some arguments in Section 9, it will be convenient to view an instance $Q_1 v_1 \ldots Q_n v_n \Psi$ of the QCSP as a two-player game between a (female) *universal* and a (male) *existential* player. The game proceeds in rounds; in round i, the players assign a value from D to the variable v_i. If Q_i is existential, the existential player chooses the value for v_i, and otherwise the universal player chooses the value. The existential player wins the game if after round n the constructed assignment satisfies all the constraints in Ψ, and otherwise the universal player wins. It is straightforward to verify that an instance of QCSP is true if and only if the existential player has a winning strategy in the corresponding game. A variable v_i is called *earlier (later)* than variable v_j in an instance $Q_1 v_1 \ldots Q_n v_n \Psi$ of the QCSP if $i < j$ $(i > j)$.

Theorem 6.2 *Let Φ be an instance QCSP(Γ) where Γ is negative. The formula Φ is false if and only if one of the following two conditions hold:*

1. *There exists a universally quantified variable that is connected in $G^{=}_{\Phi}$ to an earlier variable.*
2. *There exists a disjunction of disequalities $\psi(z_1, \ldots, z_k)$ appearing in Φ such that for all existentially quantified variables $x \in \{z_1, \ldots, z_k\}$, either*
   (a) x is connected in $G^{=}_{\Phi}$ to an earlier variable from $\{z_1, \ldots, z_k\}$, or
   (b) x is the earliest variable among the variables in $\{z_1, \ldots, z_k\}$ connected to x in G^{ψ}_{Φ}.

Proof. We first show that if one of these two conditions hold, the formula is false, by proving that the universal player has a winning strategy. If Condition 1 applies, she can play a fresh value, i.e., a value that has not been selected by any player previously in the game, at the universal variable that is connected in $G^{=}_{\Phi}$ to a previous variable, and clearly wins.

Now, suppose that Condition 2 applies to a disjunction of disequalities ψ appearing in Φ. For z from $z_1, \ldots, z_k$, let $C^{\psi}(z)$ be the connected component of z in G^{ψ}_{Φ}, and let $C^{=}(z)$ be the connected component of z in $G^{=}_{\Phi}$. Then we have: for all existential variables $x \in \{z_1, \ldots, z_k\}$, either the earliest variable in $C^{=}(x)$ is the same as the earliest variable in $C^{\psi}(x)$, or the earliest variable in $C^{=}(x)$ is a universally quantified variable. If both these conditions are violated, then the earliest variable in $C^{=}(x)$ is existential and not the earliest variable in $C^{\psi}(x)$. Thus, this earliest variable violates both (a) and (b) in Condition 2, a contradiction. Now, if the universal player has to play a value for a universally quantified variable y from $z_1, \ldots, z_k$, she plays the value of the earliest variable in $C^{\psi}(y)$ if the earliest variable has already been set; otherwise, y is the earliest variable in $C^{\psi}(y)$ and can be set arbitrarily. For all other variables she can play arbitrarily.

Suppose that there is a strategy for the existential player that wins against the above strategy for the universal player. Consider a connected component C in G^{ψ}_{Φ}. The universal variables in C will all be set to the value a of the earliest variable in C. Let x be an existential variable in C. If existential player wins, all variables in $C^{=}(x)$ must have the same value. Consider the earliest variable z in $C^{=}(x)$. By the observation of the last paragraph, z is the earliest variable in C or is universal. In the first case, we know that a is assigned to z. If z is universal, then the strategy for the universal player also assigned a to z. Hence, all values in C are mapped to a. Since this holds for all connected components C in G^{ψ}_{Φ}, the disjunction of disequalities is falsified, contradicting the assumption that existential player wins.

If Conditions 1 and 2 do not apply, then the existential player has the following winning strategy. If he has to play a value for x, and x is connected in $G^{=}_{\Phi}$ to an earlier variable, he plays the value of this earlier variable. Otherwise, he picks a fresh value. We have to show that this strategy wins against any strategy for the universal player. We first show that the constraints of the form $x = y$ in Φ are satisfied in that way. Since condition 1 does not apply, for each component of $G^{=}_{\Phi}$, all variables–except possibly the earliest–are existentially quantified. Hence, the given existential strategy satisfies all equality constraints $x = y$.

Next, consider a disjunction of disequalities ψ appearing in Φ. Since condition (2) does not hold, there exists an existentially quantified variable x of ψ such that neither (2a) nor (2b) holds. Since (2a) fails, x is (by the described strategy) set to a value different from all variables earlier than x; but, since (2b) fails, x is connected to an earlier variable z in G^{ψ}_{Φ}. Since x and z are set to different values, there exists an edge on the path from x to z that must have vertices set to different values. We claim that this edge does not correspond to an equality edge in G_Φ^ψ. Due to condition (1), the later variable (of the edge) cannot be universal; but if the later variable is existential and the edge is in G_Φ^ψ, the existential player would have set its value to the value of the earlier variable. Therefore, the edge corresponds to a disequality in ψ, and this disequality is satisfied. $\square$

Proof. (Theorem 6.1) Conditions 1 and 2 can be verified in logarithmic space, due to the result that undirected graph reachability, USTCON, is in L [24]. We have to test whether two specified vertices are connected in $G_\Phi^=$ or in G_Φ^ψ, for a disjunction of disequalities ψ. These graphs can be constructed with logarithmic work-space. The first condition needs to be tested for all pairs of variables, and the second condition for all disjunctions of disequalities, and all pairs of variables. Hence, the only non-constant additional space is needed for storing the next variables and constraints that will be tested, which is possible in logarithmic space. $\square$

Negative languages can also be characterized in terms of polymorphisms. We say that an n-ary operation $f \in O$ is *injective in the i-th argument* if for all $a, b \in D^n$ we have $f(a) \neq f(b)$ whenever $a_i \neq b_i$.

Theorem 6.3 (Proposition 68 in [3]) *Let Γ be an equality constraint language. Then Γ is negative if and only if it is preserved by all operations that are injective in one argument.*

7 Positive Languages

This section is devoted to the proof of the following.

Theorem 7.1 *Let Γ be a positive constraint language that is not negative. Then QCSP(Γ) is NP-complete.*

We begin by presenting a number of useful characterizations of positivity of constraint languages.

Definition 7.2 *We say that a formula ϕ in conjunctive normal form is* reduced*, if removing a literal or a clause from ϕ results in a formula that is not equivalent to ϕ.*

Clearly, every formula is equivalent to a formula in reduced form.

Proposition 7.3 *Let Γ be an equality constraint language over domain D. The following are equivalent:*

1. *Γ is positive, i.e., every relation in Γ is definable by a formula consisting of equalities and the connectives $\wedge$ and $\vee$.*
2. *$\mathsf{Pol}(\Gamma)$ contains all unary operations on D.*
3. *$\mathsf{Pol}(\Gamma)$ contains a non-injective unary operation having infinite image.*
4. *$\mathsf{Pol}(\Gamma)$ contains for every k a unary operation having a range of size k.*
5. *If ϕ is a reduced formula that defines a relation from Γ, then ϕ does not contain a disequality.*

Proof. (1) $\Rightarrow$ (2): Suppose that Γ is positive. Let $f : D \to D$ be any unary operation. We want to show that any relation R of Γ is preserved by f. Let $t \in R$ be any tuple. Observe that, for any two coordinates i, j, if $t_i = t_j$ then $f(t_i) = f(t_j)$. Thus, any equalities that hold in t also hold in $f(t)$, and since R has a definition that is positive in equalities, we have $f(t) \in R$.

(2) $\Rightarrow$ (3): Trivial.

(3) $\Rightarrow$ (4): A straightforward local closure argument.

(4) $\Rightarrow$ (5): Let R be an m-ary relation of Γ. Let ϕ be any reduced formula that defines R; we claim that ϕ is positive. Assume for contradiction that ϕ contains a clause ψ with a negative

literal $x_i \neq x_j$. Since ϕ is reduced, there exists an m-tuple t such that $\phi(t)$ holds and such that $x_i \neq x_j$ is the only literal that is satisfied by t in the clause ψ. Let k be the number of distinct entries of t, and let f be a unary polymorphism of Γ of range k. Then there exists a permutation α of the elements of Γ such that $f(\alpha)$ maps t_i and t_j to the same value and all other elements to distinct values. Then $f(\alpha(t))$ does not satisfy the literal $x_i \neq x_j$, and clearly also does not satisfy any other literal in ψ. Hence, ϕ is not preserved under f, a contradiction.

(5) $\Rightarrow$ (1): Trivial. □

The following known result gives the complexity upper bound on positive constraint languages.

Theorem 7.4 (follows from [20]) *For any positive constraint language Γ, the problem QCSP(Γ) is in NP.*

We mention that there is also an alternative, polymorphism-based proof of Theorem 7.4 [2].

We now show for one particular positive constraint language Γ that CSP(Γ) is NP-complete. Let P_3 be the ternary relation defined by $P_3(x, y, z) \equiv (x = y) \vee (y = z)$. Clearly, P_3 is positive.

Proposition 7.5 *The problem QCSP($(D; P_3)$) is NP-hard.*

Proof. Let L be the relation P_3 applied to the two-element domain $\{0, 1\}$ for some $0, 1 \in D$, that is, define $L = \{(a, b, c) \in \{0, 1\}^3 : (a = b) \vee (b = c)\}$. We show that the CSP over constraint language $\Gamma = (D; L, \{0\}, \{1\})$ reduces to QCSP($(D; P_3)$). Take an instance P of this CSP with variables $\{x_1, \ldots, x_n\}$. We create an instance P' of the QCSP with quantifier prefix $\forall y_0 \forall y_1 \exists x'_1 \ldots \exists x'_n$ with the following constraints:

- for each constraint $L(x_i, x_j, x_k)$ in P, we create a constraint $P_3(x'_i, x'_j, x'_k)$,
- for each constraint $\{0\}(x_i)$ in P, we create a constraint $y_0 = x'_i$,
- for each constraint $\{1\}(x_i)$ in P, we create a constraint $y_1 = x'_i$,
- for each variable $x_i \in \{x_1, \ldots, x_n\}$ in P, we create a constraint $P_3(y_0, x'_i, y_1)$.

Notice that the instance P' may have constraints of the form $y_a = x'_i$, and so might not be an instance of QCSP($(D; P_3)$). However, we can easily compute an equivalent instance P'' of QCSP($(D; P_3)$) from P', where constraints $y_a = x'_i$ are removed simply by 1) replacing all instances of x'_i in constraints with y_a, and 2) removing x'_i from the quantifier prefix.

We now argue that P' is true if and only if P is satisfiable. Suppose that P' is true. Let a_0, a_1 be two values of the domain that are not equal, and set $y_0 = a_0$ and $y_1 = a_1$. Since P' is true, there exists an assignment f' to the x'_i that make all constraints of P' true. By the last group of constraints in P', every single variable x'_i is equal to either a_0 or a_1. It is straightforward to verify that the mapping $f : \{x_1, \ldots, x_n\} \to \{0, 1\}$ where $f(x_i) = 0$ if $f'(x'_i) = a_0$, and $f(x_i) = 1$ if $f'(x'_i) = a_1$, is a satisfying assignment for P. Conversely, suppose that P is satisfiable by the assignment $f : \{x_1, \ldots, x_n\} \to \{0, 1\}$. Then, for any assignment to the universal variables y_0, y_1, the assignment to the x'_i that sets x'_i to $y_{f(x_i)}$ satisfies the constraints of P'.

Finally, we wish to show that the CSP over Γ is NP-hard. By [17], it suffices to show that Γ does not have as polymorphism one of the operations $\{0, 1, \wedge, \vee, m, a\}$ where 0 and 1 are the constant operations, $\wedge$ and $\vee$ are the boolean AND and OR operations, respectively, $m : \{0, 1\}^3 \to \{0, 1\}$ is the majority operation $m(x, y, z) = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$, and $a : \{0, 1\}^3 \to \{0, 1\}$ is the affine operation $a(x, y, z) = x \oplus y \oplus z$. Clearly, Γ does not have either of the constants 0, 1 as polymorphism: 1 does not preserve $\{0\}$, and 0 does not preserve $\{1\}$. The operation $\wedge$ does not preserve P_3 because $(1, 1, 0) \wedge (0, 1, 1) = (0, 1, 0)$. The operation $\vee$ does not preserve P_3 because $(0, 0, 1) \vee (1, 0, 0) = (1, 0, 1)$. The operation m does not preserve P_3 because $m((1, 1, 0), (0, 1, 1), (0, 0, 0)) = (0, 1, 0)$. And, the operation a does not preserve P_3 because $a((1, 1, 0), (0, 1, 1), (0, 0, 0)) = (1, 0, 1)$. □

Theorem 7.6 (From [3], Proposition 37) *Let f be an essential operation with infinite image. Then f preserves $\neq$, or it generates all operations.*

We can now prove Theorem 7.1.

Proof. The containment of QCSP(Γ) in NP is given by Theorem 7.4. If $\mathsf{sPol}(\Gamma)$ contains an essential operation, then Γ is preserved by all operations by Theorem 7.6. In particular, every relation in Γ is negative, by Theorem 6.3, in contradiction to our assumptions.

So we can assume that all operations in $\mathsf{sPol}(\Gamma)$ are essentially unary. We prove NP-hardness of QCSP(Γ) as follows. Essentially unary operations clearly preserve the relation $P_3 \equiv (x = y) \vee (y = z)$. Corollary 4.4 then shows that P_3 is ∀∃∧-definable in Γ. By Lemma 3.1, the problem QCSP((D, P_3)), which is NP-hard by Proposition 7.5, reduces to QCSP(Γ). □

8 Analysis of Non-Positive Languages

We have already proved the classification theorem for positive languages. We now show that if a language is not positive, then it is either negative or it can ∀∃∧-define the relation I defined by

$$I(x, y, z) \equiv ((x = y) \Rightarrow (y = z)) .$$

In Section 9, we finally show the QCSP for the relation I is coNP-hard.

Theorem 8.1 *Let Γ be a constraint language that is neither positive nor negative. Then* [Γ] *contains I.*

We will use the following results from [3] and from [4].

Theorem 8.2 (Lemma 38 in [3]) *Any operation with infinite range that violates ≠ generates a unary non-injective function with infinite range.*

Theorem 8.3 (of [4]) *Let g be an essential operation. If g preserves ≠, then it generates all injective operations.*

It was shown in [4] that if Γ is preserved by all injective operations, then all relations of Γ can be defined by (quantifier-free) *Horn formulas* over $(D; =)$, i.e., by quantifier-free formulas in conjunctive normal form where each clause contains at most one positive literal. It will later be convenient to have the following slight strengthening of this result.

Proposition 8.4 *If ϕ is a reduced definition of a relation preserved by a binary injective operation g, then ϕ is a Horn formula.*

Proof. Let ϕ be a reduced formula that defines R (Definition 7.2). We show that no clause in ϕ contains two equalities. Assume otherwise that there exists a clause ψ of ϕ which contains two equalities $x_s = x_t$ and $x_i = x_j$. Construct ϕ' from ϕ by removing the equation $x_s = x_t$, and ϕ'' by removing $x_i = x_j$. There exist $a, b \in D^n$ such that $\phi(a)$ but not $\phi'(a)$, and $\phi(b)$ but not $\phi''(b)$. Clearly, $a_s = a_t$, $a_i \neq a_j$, $b_s \neq b_t$, and $b_i = b_j$. Because R is preserved by g, it contains a tuple $c = g(a, b)$ where $c_s \neq c_t$ and $c_i \neq c_j$. But then $\phi(c)$ does not hold, a contradiction. □

It will be helpful to consider the relation

$$S = \{(x, y, z) \mid (x = y \wedge y = z) \vee (x \neq y \wedge y \neq z \wedge x \neq z)\}$$

containing all tuples whose values are either all equal, or all different, and the relation

$$T = \{(a, b, c, d) \mid a = b \neq c = d \text{ or the values } a, b, c, d \text{ are pairwise distinct}\} .$$

Note that both relations consist of exactly two orbits of tuples. This will be relevant due to the following general lemma (which holds for arbitrary structures).

Lemma 8.5 *Let R be a k-ary relation that consists of m orbits of k-tuples. Then every operation f that violates R generates an at most m-ary operation that violates R.*

Proof. Let f' be an operation of smallest arity l that is generated by f and that violates R. Then there are k-tuples $t_1, \dots, t_l \in R$ such that $f'(t_1, \dots, t_l) \notin R$. For $l > m$ there are two tuples t_i and t_j that lie in the same orbit of k-tuples, and therefore there is a permutation h such that $h(t_j) = t_i$. By permuting the arguments of f', we can assume that $i = 1$ and $j = 2$. Then the $(l-1)$-ary operation g defined as $g(x_2, \dots, x_l) := f'(h(x_2), x_2, \dots, x_l)$ also violates R, a contradiction. Hence, $l \leq m$. □

Lemma 8.6 *The relation S can pp-define the relation I.*

Proof.

$$I(u, v, w) \equiv \exists x \exists x' (S(u, v, x) \wedge S(u, v, x') \wedge S(x, x', w))$$

is a pp-definition of I.

We verify this as follows. Consider the pp-definition. If u and v are equal, then by the predicates $S(u, v, x)$ and $S(u, v, x')$, we must have that $u = v = x = x'$, and then by the predicate $S(x, x', w)$, it must hold that $u = v = w$. Moreover, if u, v, w are all equal, then setting x and x' to the same value as that of u, v, w, we have that all three predicates are satisfied.

If u and v are not equal, then for any assignment to w, if x and x' are set to values that are pairwise distinct from each other and from each of u, v, w, then all three of the predicates are satisfied. □

Lemma 8.7 *The relation T can pp-define the relation I.*

Proof. By Lemma 8.6, it suffices to show that T expresses the relation S of Lemma 8.6. We claim that S has the following pp-definition.

$$S(u, v, x) \equiv \exists y \exists z (T(u, v, y, z) \wedge T(v, x, y, z) \wedge T(u, x, y, z)) \; .$$

We verify this as follows. If $u = v = x$ all have the same value then setting $y = z$ to be a distinct value results in all of the predicates being satisfied. If u, v, and x have pairwise distinct values then setting y and z to have values pairwise distinct from each other and from the values of u, v, x, also results in both of the predicates being satisfied.

We now prove that all other tuples do not satisfy the given pp-formula. Suppose that $u = v$. Then it must hold, by the first predicate $T(u, v, y, z)$, that $y = z$. This in turn implies, by the second predicate $T(v, x, y, z)$, that $v = x$. Thus $u = v = x$. Next, suppose that $u \neq v$. By the first predicate, it must hold that u, v, y, z are all pairwise distinct, and then the second predicate implies that v, x, y, z are all pairwise distinct (in particular, $x \neq v$), so we have by the third predicate that $x \neq u$, that is, u, v, x are pairwise distinct. □

Lemma 8.8 *If R is not negative and preserved by a binary injective operation, then the relation I has a pp-definition in $(D; R, \neq)$.*

Proof. Let ϕ be a reduced definition of R. Proposition 8.4 shows that ϕ is a Horn formula. Among all reduced definitions, we choose one that minimizes the vector telling us how many clauses there are of each length, in a lexicographical sense: we prefer any number of length 3 clauses to a length 4 clause, for instance. Since R is Horn but not negative, ϕ contains a clause C of the form $x_{i_1} = x_{j_1} \vee x_{i_2} \neq x_{j_2} \vee \cdots \vee x_{i_k} \neq x_{j_k}$ for $k \geq 2$.

Because ϕ is reduced, we know we have some tuple $t \in R$ where $x_{i_1} = x_{j_1}$ is true and all other literals in C are false, and we also have some tuple $s \in R$ where $x_{i_2} \neq x_{j_2}$ is true and all other literals in C are false. Let $R'(x_1, \dots, x_n)$ be the relation defined by the formula

$$\phi(x_1, \dots, x_n) \wedge \bigwedge_{l=3}^{k} x_{i_l} = x_{j_l} \; ,$$

and let R'' be the projection of R' to the i_1th, j_1th, i_2th, and j_2th argument. Note that $|\{i_1, j_1, i_2, j_2\}| \geq 3$ because ϕ is reduced, and therefore the arity of R'' is either 3 or 4. Because $t \in R$ satisfies $x_{i_1} = x_{j_1}$ and $x_{i_2} = x_{j_2}$, and also satisfies all the additional equations

$x_{i_l} = x_{j_l}$ for $l \geq 3$, the relation R'' contains a tuple t' that satisfies $x_{i_1} = x_{j_1}$ and $x_{i_2} = x_{j_2}$. The analogous argument for s shows that R'' contains a tuple s' where $x_{i_1} \neq x_{j_1}$ and $x_{j_1} \neq x_{j_2}$.

First, consider the case that $|\{i_1, j_1, i_2, j_2\}| = 3$. By renaming variables, we can assume that $j_1 = i_2$. Suppose that R'' contains a triple with three distinct entries. We know that $R''(x_{i_1}, x_{j_1}, x_{j_2})$ does not contain a tuple of the form (x, x, y) for $x \neq y$, because such a tuple violates $C \wedge \bigwedge_{l=3}^{k} x_{i_l} = x_{j_l}$. But then the relation defined by $R''(x, y, z) \wedge R''(y, z, x) \wedge R''(z, x, y)$ only contains the tuple with three distinct entries and the tuple with three equal entries, and hence we have found a pp-definition of the relation S in $(D; R, \neq)$. By Lemma 8.6 we are done.

Suppose now that $R''(x_{i_1}, x_{i_2}, x_{j_2})$ does not contain a tuple with three distinct entries. Then it is impossible that there is a tuple $r \in R''$ where $x_{i_1} \neq x_{j_2}$, because the tuple obtained from applying the binary injective operation to r and s' has three distinct entries. Thus the clause $x_{i_3} \neq x_{j_3} \vee \cdots \vee x_{i_k} \neq x_{j_k} \vee x_{i_1} = x_{j_2}$ is entailed by ϕ. Hence, we could replace the clause C in ϕ by this shorter clause and obtain a formula that is equivalent to ϕ, because $x_{i_1} = x_{j_2}$ also implies $x_{i_1} = x_{j_1} \vee x_{j_1} \neq x_{j_2}$. This contradicts the choice of ϕ.

Now take the case where $|\{i_1, j_1, i_2, j_2\}| = 4$. Observe that R'' does not contain tuples with $x_{i_1} \neq x_{j_1}$ and $x_{i_2} = x_{j_2}$, because such tuples violate $C \wedge \bigwedge_{l=3}^{k} x_{i_l} = x_{j_l}$.

If $t' \in R''$ satisfies $x_{i_1} = x_{j_1} \neq x_{i_2} = x_{j_2}$, then the tuple obtained from applying the binary injective operation to t' and s' has four distinct entries. It is easy to verify that the formula

$$R''(a, b, c, d) \wedge R''(c, d, a, b) \wedge a \neq c \wedge a \neq d \wedge b \neq c \wedge b \neq d \qquad (1)$$

is a pp-definition of the relation $T(a, b, c, d)$, and we are done by Lemma 8.7.

Now suppose that $t' \in R''$ satisfies $x_{j_1} = x_{i_2}$ and hence t' is the tuple with four equal entries. If the tuple in R'' that has the most values has exactly two values, by the observation above each of these values must occur twice. So we have two forced equalities $y = z$ and $x = w$, and we can observe as before that $x_{i_3} \neq x_{j_3} \vee \cdots \vee x_{i_k} \neq x_{j_k} \vee y = z$ and $x_{i_3} \neq x_{j_3} \vee \cdots \vee x_{i_k} \neq x_{j_k} \vee x = w$ are both entailed by the original formula, and we could replace the original clause with these two clauses.

So suppose that the tuple in R'' that has the most values only has one value that occurs twice. We can suppose this tuple has the form (a, b, b, c), up to rearranging coordinates. We claim that the expression

$$R''(a, b, b, c) \wedge R''(c, a, a, b) \wedge R''(b, c, c, a)$$

defines the relation $S(a, b, c)$. Clearly, the tuple that contains three distinct values, and the tuple that contains three equal values satisfy the expression. For any other tuple there is a conjunct in the expression that has the form $R''(a, b, b, b)$ for $a \neq b$, and by the observation above the tuple is excluded.

Finally, suppose that the tuple in R'' that has the most values contains four distinct values. We know that some tuple where one value occurs once and another value occurs three times is excluded (we have just shown this). If some tuple where two values occur twice is excluded from R'', then the relation

$$\exists a \bigwedge_{\pi} R''(\pi(a), \pi(b), \pi(c), \pi(d))$$

is a pp-definition of the relation $S(b, c, d)$, where the conjunction is over all permutations π of $\{a, b, c, d\}$. Otherwise R'' contains all tuples where two values occur exactly twice. As before it is easy to verify that Expression (1) is a pp-definition of $T(a, b, c, d)$, and we are again done by Lemma 8.7. $\square$

Proof. (Theorem 8.1) Let Γ be a constraint language that is neither positive nor negative. If an operation from $\mathsf{sPol}(\Gamma)$ violates $\neq$ then Theorem 8.2 and Proposition 7.3 show that Γ is positive, a contradiction to our assumptions. So we can assume without loss of generality (by Corollary 4.4) that Γ contains the relation $\neq$.

If all operations in $\mathsf{sPol}(\Gamma)$ are essentially unary, then they all preserve I. In this case I has a $\forall\exists\wedge$-definition in Γ and we are done. If $\mathsf{sPol}(\Gamma)$ contains an essential operation, then Theorem 8.3 implies that Γ is preserved by a binary injection.

Since Γ is not negative, Γ contains a relation R without a negative definition. We can therefore apply Lemma 8.8 and have that I is pp-definable in Γ. $\square$

9 coNP-hardness

In this section, we prove that the QCSP over the relation $I = \{(x, y, z) \mid (x = y \Rightarrow y = z)\}$ is coNP-hard.

Theorem 9.1 *QCSP$((D; I))$ is coNP-hard.*

Proof. We reduce from the complement of the problem MONOTONE 3-SAT. In this version of the 3-SAT problem, the input is a 3-SAT instance where all clauses are either purely negative or purely positive; MONOTONE 3-SAT is known to be NP-hard [11].

Let ϕ be an instance of the MONOTONE 3-SAT problem, use $v_1, \ldots, v_n$ to denote the variables, use $N_1, \ldots, N_l$ to denote the negative clauses, and use $P_1, \ldots, P_m$ to denote the positive clauses. From this instance, we create an instance ψ of the problem QCSP$((D; I))$ as follows. The instance ψ has quantifier prefix

$$\exists b_0 \exists b_1 \forall v_1 \ldots \forall v_n \exists N_1 \ldots \exists N_l \exists N_2' \ldots \exists N_l' \exists P_1 \ldots \exists P_m \exists P_2' \ldots \exists P_m'.$$

The quantifier-free part of ψ is the conjunction of the following:

1. $b_0 \neq b_1$. Note that we are free to use $\neq$, as $u \neq v$ is definable by $\forall w(u = v \Rightarrow v = w)$.
2. $(v_i = b_0 \Rightarrow b_0 = N_h) \wedge (v_j = b_0 \Rightarrow b_0 = N_h) \wedge (v_k = b_0 \Rightarrow b_0 = N_h)$ for each negative clause $N_h = (\neg v_i \vee \neg v_j \vee \neg v_k)$.
3. $(v_i = b_1 \Rightarrow b_1 = P_h) \wedge (v_j = b_1 \Rightarrow b_1 = P_h) \wedge (v_k = b_1 \Rightarrow b_1 = P_h)$ for each positive clause $P_h = (v_i \vee v_j \vee v_k)$.
4. $(N_1 = N_2 \Rightarrow N_2 = N_2') \wedge \bigwedge_{h=3,\ldots,l}(N_{h-1}' = N_h \Rightarrow N_h = N_h')$
5. $(P_1 = P_2 \Rightarrow P_2 = P_2') \wedge \bigwedge_{h=3,\ldots,m}(P_{h-1}' = P_h \Rightarrow P_h = P_h')$
6. $N_l' = P_m'$. Note that we are free to use $=$ since $u = v$ is by definition a quantified constraint formula.

(We assume that the formula ϕ contains at least two positive clauses and two negative clauses.)

We verify the correctness of this reduction as follows. First, assume that the original formula ϕ is satisfiable via the assignment $f : \{v_1, \ldots, v_n\} \to \{0, 1\}$. Consider the induced assignment to the universally quantified variables of ψ where the universally quantified variable v_i is set to be $b_{f(v_i)}$. Since f satisfies all of the clauses of ϕ, in order for all constraints in (2), (3) of ψ to be satisfied, it must hold that $N_h = b_0$ for all $h \in [l]$ and $P_h = b_1$ for all $h \in [m]$. For all constraints in (4), (5) to be satisfied, it must further hold that $N_h' = b_0$ for all $h = 2, \ldots, l$ and that $P_h' = b_1$ for all $h = 2, \ldots, m$. By (1), this implies that $N_l' \neq P_m'$, contradicting (6), and we have that the formula ψ is false.

Now, assume that the original formula ϕ is not satisfiable. Consider an assignment $g : \{v_1, \ldots, v_n\} \to D$ to the universally quantified variables of ψ. Let us say that a negative clause N_h is not satisfied by g if none of its variables are set by g to be b_0; similarly, let us say that a positive clause P_h is not satisfied by g if none of its variables are set by g to be b_1. By assumption, there exists a negative clause not satisfied by g or a positive clause not satisfied by g. Consider the case where a negative clause N_h is not satisfied by g (the case of a positive clause is similar). All of the constraints in (2) can be satisfied in such a way that the variable N_h is not set equal to b_0. It follows that all of the constraints in (4) can be satisfied in such a way that the variable N_l' is set equal to b_1. Setting all of the variables P_h and P_h' equal to b_1 results in all of the constraints being satisfied. We conclude that the formula ψ is true. $\square$

10 Concluding Remarks

Figure 1 is an illustration of the complexity landscape of equality constraint languages.

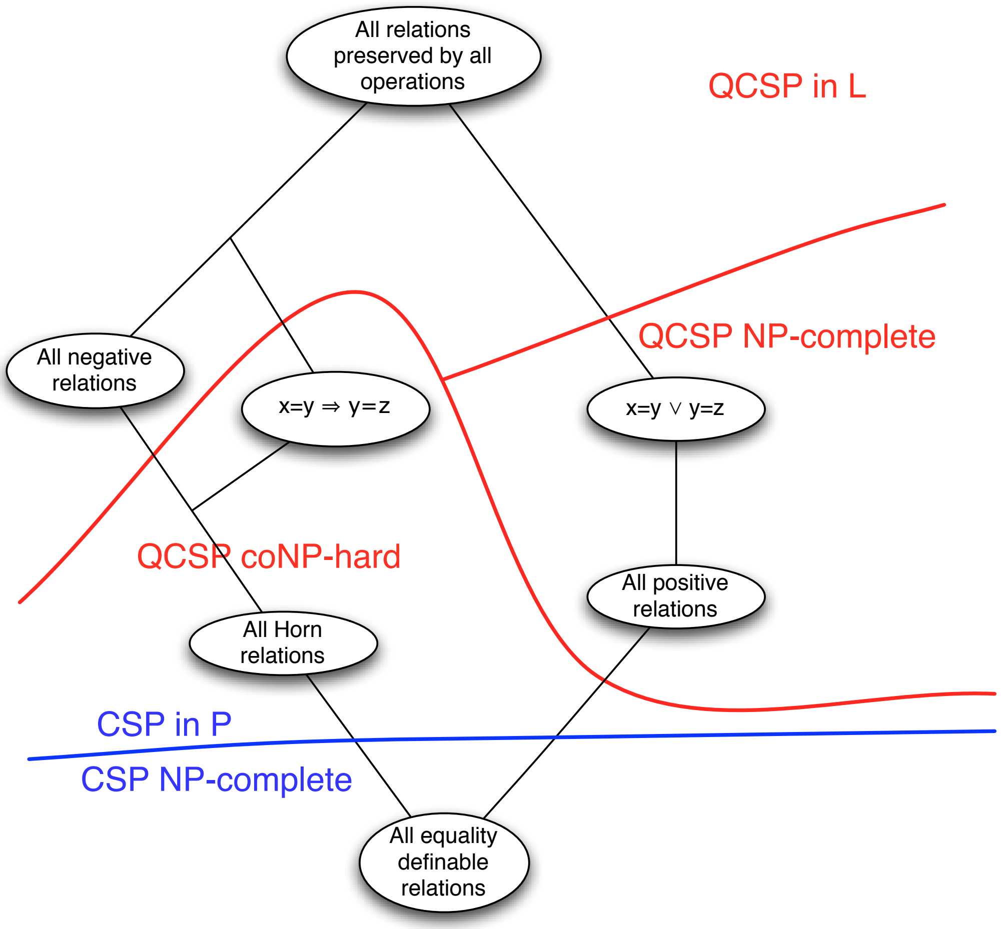

Figure 1: The complexity of the quantified constraint satisfaction problem for equality constraint languages.

We remark that the classification of the computational complexity of QCSP(Γ) for equality constraint languages Γ presented in this paper is *effective* in the sense that there is an algorithm that decides in a finite amount of time whether a given constraint language (whose relations are represented by equality-formulas) has a QCSP in L or has a QCSP that is coNP-hard or NP-hard. This is obvious from the syntactic descriptions of the corresponding equality constraint languages presented in the classification theorem.

We did not determine the precise complexity of the coNP-hard QCSPs for equality templates, and suspect that all of them are in fact PSPACE-complete. In the conference version of the paper, we presented a false proof of this. Proving PSPACE-hardness for the problem QCSP((D, I)) appears to be very difficult, and we have to leave this open here.

Acknowledgements. We thank the referees for their helpful comments; one of the comments led to the discovery of the mistake in the PSPACE-hardness proof of the conference version of the paper.

References

[1] L. Barto and M. Kozik. Constraint satisfaction problems of bounded width. In *Proceedings of Symposium on Foundations of Computer Science (FOCS)*, pages 595–603, 2009.

[2] M. Bodirsky and H. Chen. Collapsibility in infinite-domain quantified constraint satisfaction. In *Proceedings of Computer Science Logic (CSL), Szeged, Hungary*, 2006.

[3] M. Bodirsky, H. Chen, and M. Pinsker. The reducts of equality up to primitive positive interdefinability. *To appear in the Journal of Symbolic Logic*, 2010.

[4] M. Bodirsky and J. Kára. The complexity of equality constraint languages. *Theory of Computing Systems*, 3(2):136–158, 2008. A conference version appeared in the proceedings of CSR'06.

[5] M. Bodirsky and J. Nešetřil. Constraint satisfaction with countable homogeneous templates. *Journal of Logic and Computation*, 16(3):359–373, 2006.

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

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

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

[9] A. A. Bulatov and V. Dalmau. A simple algorithm for Mal'tsev constraints. *SIAM J. Comput.*, 36(1):16–27, 2006.

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

[11] M. Garey and D. Johnson. *A guide to NP-completeness.* CSLI Press, Stanford, 1978.

[12] L. Haddad and I. G. Rosenberg. Finite clones containing all permutations. *Canad. J. Math.*, 46(5):951–970, 1994.

[13] P. Hell and J. Nešetřil. *Graphs and Homomorphisms.* Oxford University Press, Oxford, 2004.

[14] W. Hodges. *A shorter model theory.* Cambridge University Press, Cambridge, 1997.

[15] P. M. Idziak, P. Markovic, R. McKenzie, M. Valeriote, and R. Willard. Tractability and learnability arising from algebras with few subpowers. In *Proceedings of LICS'07*, pages 213–224, 2007.

[16] P. Jeavons, D. Cohen, and M. Gyssens. Closure properties of constraints. *JACM*, 44(4):527–548, 1997.

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

[18] J. Keisler. Reduced products and Horn classes. *Trans. Amer. Math. Soc.*, 117:307–328, 1965.

[19] E. W. Kiss and M. Valeriote. On tractability and congruence distributivity. In *Proceedings of LICS'06*, pages 221–230, 2006.

[20] D. Kozen. Positive first-order logic is NP-complete. *IBM Journal of Research and Development*, 25(4):327–332, 1981.

[21] A. I. Mal'tsev. A strengthening of the theorems of Slupecki and Iablonskii (Russian, english summary). *Algebra i Logika (3)*, 6:61–75, 1967.

[22] D. Marker. *Model Theory: An Introduction.* Springer, New York, 2002.

[23] M. Maróti and R. McKenzie. Existence theorems for weakly symmetric operations. *Algebra Universalis*, 59(3-4):463489, 2008.

[24] O. Reingold. Undirected st-connectivity in log-space. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 376–385, 2005.

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

[26] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time: Preliminary report. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 1–9, 1973.

[27] A. Szendrei. *Clones in universal Algebra.* Séminaire de Mathématiques Supérieures. Les Presses de L'Université de Montréal, 1986.