

Quantified Equality Constraints

Manuel Bodirsky
Institut für Informatik
Humboldt-Universität zu Berlin, Germany
bodirsky@informatik.hu-berlin.de

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

Abstract

An equality template (also equality constraint language) is a relational structure with infinite universe whose relations can be defined by boolean combinations of equalities. We prove a complete complexity classification for quantified constraint satisfaction problems (QCSPs) over equality templates: these problems are in L (decidable in logarithmic space), NP-complete, or PSPACE-complete. 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

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

over a finite relational signature, relative to a relational structure over the same signature. For a relational structure Γ , define $\text{CSP}(\Gamma)$ to be the CSP where the relational structure is fixed to be Γ . Note that the instances of $\text{CSP}(\Gamma)$ are syntactic objects, namely, the pp-sentences over the signature of Γ . When discussing problems of the form $\text{CSP}(\Gamma)$, we call Γ the *template*. In a now oft-cited result, Schaefer [20] studied the family of problems $\text{CSP}(\Gamma)$ over templates with two-element universes. He proved a *dichotomy theorem*: every such problem $\text{CSP}(\Gamma)$ is either in P, or is NP-complete. In recent years, there has been considerable research effort directed towards classifying the complexity of $\text{CSP}(\Gamma)$ over all finite templates Γ [11, 7, 5, 6, 14, 12, 10].

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 $\text{CSP}(\Gamma)$ is known to be NP-hard. Another feature of his (and subsequent) work is that it places tractable cases of $\text{CSP}(\Gamma)$ into a unified framework; the tractable cases of $\text{CSP}(\Gamma)$ encompassed by Schaefer's theorem include 2-SAT and HORN SAT, two classic tractable fragments of propositional logic.

In recent work, Bodirsky and Kara [2] gave what appears to be the first systematic $\text{CSP}(\Gamma)$ classification result for templates Γ over an *infinite* universe. They studied the natural and basic 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 $\text{CSP}(\Gamma)$ is either in P or NP-complete.

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 complete complexity classification for the problems $\text{QCSP}(\Gamma)$ over equality templates, showing that each such problem is either in L (decidable in logarithmic space), NP-complete, or PSPACE-complete. 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 as well as other PSPACE problems. 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 [16] showing that *positive* first-order sentences have an NP-complete validity problem. To the best of our knowledge, our positive result showing certain formulas to be de-

cidable 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 $x_1 \neq y_1 \vee \dots \vee x_k \neq y_k$. 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 $\text{QCSP}(\Gamma)$ is in L.

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 $\text{QCSP}(\Gamma_1)$ 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 is easy to verify that the formula above is true in Γ_1 . Since Γ_1 is negative, it follows from our classification theorem that $\text{QCSP}(\Gamma_1)$ 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 $\text{QCSP}(\Gamma)$ 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. It can be verified that Γ_2 is not negative, and so our classification theorem implies that $\text{QCSP}(\Gamma_2)$ 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 $\text{QCSP}(\Gamma)$ that is PSPACE-complete.

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 negative nor positive, and hence, by our classification theorem, the problem $\text{QCSP}(\Gamma_3)$ is PSPACE-complete.

Techniques used. As we mentioned, the complexity of the CSP over equality templates has been classified recently [2]. This result was obtained by a universal-algebraic approach which was originally developed for studying the CSP over finite domains [7, 5, 6, 14, 12, 10]. The approach rests on studying certain closure properties, so-called *polymorphisms*, that a template has. 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 [2] 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 this paper, we also present a *surjective preservation theorem* which concerns the expressibility of a template 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 ω -categorical template Γ if and only if it is preserved by all *surjective* polymorphisms of Γ . This theorem was previously shown for *finite* structures [4]. Our classification result is, to the best of our knowledge, the first result on QCSP complexity that is based on such a surjective preservation theorem. The proof that we give here relies on known preservation theorems in model theory [15].

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 [13, 17] and [22].

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 of arity k_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 .

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

Countable Categoricity. An important class of structures for constraint satisfaction is the class of ω -categorical structures, defined as follows.

Definition 2.1 *A countable relational structure Γ is called ω -categorical, if all countable models of $\text{Th}(\Gamma)$ are isomorphic to Γ .*

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 a permutation on D . The automorphism group of Γ is called *oligomorphic*, if it has a finite number of orbits of k -tuples, for each $k \geq 1$. Many techniques known for constraint satisfaction with finite templates can be generalized to constraint satisfaction with ω -categorical templates; essentially, this is due to the following theorem; see [8].

Theorem 2.2 (Engeler, Ryll-Nardzewski, Svenonius)

Let Γ be a relational structure. Then Γ is ω -categorical if and only if the automorphism group of Γ is oligomorphic.

Homomorphisms. Let Γ and Γ' be τ -structures. 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 .

Polymorphisms. Let D be a countable set, and 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 $R(x_1^i, \dots, x_m^i)$ holds for all $1 \leq i \leq k$ in Γ , then $R(f(x_1^1, \dots, x_1^k), \dots, f(x_m^1, \dots, x_m^k))$ 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 $\Gamma^k = \Gamma \times \dots \times \Gamma$ to Γ , where $\Gamma_1 \times \Gamma_2$ is the (*direct*) *product* of the two relational τ -structures Γ_1 and Γ_2 ([13]; this product is also called the *categorical product* or the *cross product* [12]). Hence, the unary bijective polymorphisms are the automorphisms of Γ . We use $\text{Pol}(\Gamma)$ to denote the polymorphisms of Γ , and $\text{sPol}(\Gamma)$ to denote the surjective polymorphisms of Γ .

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

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

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. If f is in the smallest clone that contains a set of operations F , we say that f is *generated* by F .

It is easy to verify that the set $\text{Pol}(\Gamma)$ of all polymorphisms of Γ is a clone with domain D_Γ . Moreover, $\text{Pol}(\Gamma)$ is also *locally closed*, a property that is defined as follows. We say that an operation f is *interpolated* by a set F 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 only study clones on a countably infinite domain D that contain all permutations of D . We therefore slightly abuse notation 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 .

An operation f is *essentially unary* if there is a unary operation f_0 such that $f(x_1, \dots, x_k) = f_0(x_i)$ for some fixed $i \in \{1, \dots, k\}$. We say that a k -ary operation f *depends on argument i* if there is no $k-1$ -ary operation f' such that $f(x_1, \dots, x_k) = f'(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_k)$. We can equivalently characterize k -ary operations that depend on the i -th argument by requiring that there are elements x_1, \dots, x_k and x'_i such that $f(x_1, \dots, x_k) \neq f(x_1, \dots, x_{i-1}, x'_i, x_{i+1}, \dots, x_k)$. 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 also 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. We now identify a simple but useful lemma.

Lemma 2.3 *A binary operation is essential if and only if there are values a, a', b, b' such that $f(a, b) \neq f(a', b)$ and $f(a, b) \neq f(a, b')$.*

3 Quantified Constraint Satisfaction

A *constraint language* is simply a relational structure; we typically refer to a relational structure Γ with signature τ as a constraint language when we are interested in the computational problem $\text{QCSP}(\Gamma)$, which is defined below. 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 $\text{QCSP}(\Gamma)$, 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 . 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 (see for example [13]). We therefore focus in the rest of the paper on constraint languages Γ with a countably infinite domain.

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*) 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 set of all relations that are $\forall\exists\wedge$ -definable from Γ , and we use $\langle\Gamma\rangle$ to denote the set of all relations that are pp-definable from Γ .

In the following, we frequently make use of the following result, which was shown in [4] 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 $\text{QCSP}(\Gamma_1)$ is logarithmic space reducible to $\text{QCSP}(\Gamma_2)$.*

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. For the QCSP of equality constraint languages, it is not important which countably infinite domain we use, and we will generally use the symbol D to denote a countably infinite domain. All equality constraint languages admit quantifier elimination [13]: this is, all equality-formulas are logically equivalent to a quantifier-free equality-formula. Therefore, equality constraint languages can be defined equivalently as those relational structures where all relations can be defined by a boolean combination of atoms of the form $x = y$.

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 . This group is oligomorphic. (From the fact that the automorphism group of an equality constraint language is oligomorphic, it follows by Theorem 2.2 that any equality constraint language is ω -categorical.) It is also straightforward to verify that all injective unary operations on D are polymorphisms of an equality constraint language Γ on D . Observe that, if a tuple $t = (t_1, \dots, t_k)$ is an element of an equality-definable relation $R \subseteq D^k$, then the orbit of t is also contained in R .

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 [21].

Proposition 3.2 *For every equality constraint language Γ , the problem $\text{QCSP}(\Gamma)$ is in PSPACE.*

4 The Surjective Preservation Theorem

It was proved in [4] 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 . We show that the same characterization holds for ω -categorical structures Γ (Theorem 4.2). Our proof relies on the following model-theoretic preservation theorem, which follows from [15] (the result is not covered by standard text books as [9] or [13, 17]). We thank J. Keisler for helpful explanations.

Theorem 4.1 (essentially from [15]) *A formula ϕ is equivalent to a $\forall\exists\wedge$ -formula modulo a first-order theory T if ϕ is preserved by surjective homomorphisms from finite direct products of models of T to models of T .*

A standard model-theoretic argument allows one to deduce the desired preservation theorem for surjective polymorphisms.

Theorem 4.2 *Let Γ be an ω -categorical structure. Then a relation R has a $\forall\exists\wedge$ -definition in Γ if and only if R is preserved by all surjective polymorphisms of Γ .*

Corollary 4.3 *Let Γ_1, Γ_2 be ω -categorical constraint languages. If $\text{sPol}(\Gamma_2) \subseteq \text{sPol}(\Gamma_1)$, then $\text{QCSP}(\Gamma_1)$ is logarithmic space reducible to $\text{QCSP}(\Gamma_2)$.*

Proof. Directly from Lemma 3.1 and Theorem 4.2. \square

Note that Corollary 4.3 holds in particular for equality constraint languages.

5 Classification Theorem

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

The first property is that of being *negative*.

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 property is that of being *positive*.

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 proposition gives useful characterizations of positivity of constraint languages. Its proof anticipates concepts and arguments from a forthcoming paper on the lattice of locally closed clones that contain all permutations [1]. The proof for the implication from 6 to 3 is due to Michael Pinsker.

Definition 5.5 *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 5.6 *Let Γ be an equality constraint language over domain D . The following are equivalent:*

1. Γ is positive
2. $\text{Pol}(\Gamma)$ contains all unary operations on D
3. $\text{Pol}(\Gamma)$ contains a non-injective unary operation having infinite image
4. $\text{sPol}(\Gamma)$ contains a non-injective unary operation
5. $[\Gamma]$ does not contain \neq
6. $\text{sPol}(\Gamma)$ contains an operation that generates a constant operation.

The following is the statement of our classification theorem.

Theorem 5.7 (Classification theorem) *Let Γ be an equality constraint language.*

- If Γ is negative, then the problem $\text{QCSP}(\Gamma)$ is decidable in logarithmic space.
- Else, if Γ is positive, then the problem $\text{QCSP}(\Gamma)$ is NP-complete.
- Else, the problem $\text{QCSP}(\Gamma)$ is PSPACE-complete.

In the second and third cases, the problem $\text{QCSP}(\Gamma)$ 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 $\text{QCSP}(\Gamma)$ is decidable in logarithmic space. Otherwise, and if the constraint language is positive, then Theorem 7.1 shows that $\text{QCSP}(\Gamma)$ is NP-complete. If the constraint language is not positive, then Theorem 8.1 shows that either Γ is negative, and we are again done, or $[\Gamma]$ contains the relation I (defined in Section 8). It is easy to verify that I is neither negative nor positive. By Theorem 9.1, we have that $\text{QCSP}((D; I))$ is PSPACE-hard; $\text{QCSP}((D; I))$ reduces to $\text{QCSP}(\Gamma)$ 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 $QCSP(\Gamma)$ is decidable in logarithmic space.*

Let Γ be negative, and let Φ be an instance of $QCSP(\Gamma)$ with variables $X = \{x_1, \dots, x_n\}$. If $R(v_1, \dots, v_k)$ is a constraint in Φ , we use $\phi_{R(v_1, \dots, v_k)}$ to denote the formula with variables v_1, \dots, 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 $\phi_{R(\bar{v})}$ with $R(\bar{v})$ a constraint of Φ . We will make use of the following graphs.

- The undirected graph $G_{\Phi}^{\bar{=}}$ is defined to have 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 $R(\bar{v})$ in Φ such that $\phi_{R(\bar{v})}$ 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 $G_{\Phi}^{\bar{=}}$ 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 the graphs $G_{\Phi}^{\bar{=}}$ and G_{Φ}^{ψ} omit paths of certain types; otherwise, it is false.

For the correctness proof of the following algorithm, but also for some arguments in Section 9, it will be convenient to view an instance $Q_1 v_1 \dots Q_n v_n \Psi$ of the $QCSP$ as a two-player game between a *universal* and an *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 the $QCSP$ 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 $Q_1 v_1 \dots Q_n v_n \Psi$ of the $QCSP$ if $i < j$ ($i > j$).

Theorem 6.2 *Let Φ be an instance of $QCSP(\Gamma)$ 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 $G_{\Phi}^{\bar{=}}$ to an earlier variable.*

2. *There exists a disjunction of disequalities $\psi(z_1, \dots, z_k)$ appearing in Φ such that for all existentially quantified variables $x \in \{z_1, \dots, z_k\}$, either*

- (a) *x is connected in $G_{\Phi}^{\bar{=}}$ to an earlier variable from $\{z_1, \dots, z_k\}$, or*
- (b) *x is the earliest variable among the variables in $\{z_1, \dots, z_k\}$ that are connected to x in G_{Φ}^{ψ} .*

Proof. (Theorem 6.1) We use the recent result that $L=SL$ [19]. From this result, a problem is in L if it can be Turing-reduced to undirected graph reachability, $USTCON$. The undirected connectivity queries that we are going to use will be exclusively with respect to graphs of the form $G_{\Phi}^{\bar{=}}$ or G_{Φ}^{ψ} where ψ is a disjunction of disequalities appearing in Φ .

Condition 1 and 2 can be verified in logarithmic space, once we have an oracle that decides whether two specified vertices are connected in $G_{\Phi}^{\bar{=}}$ 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

We want to remark that every constraint language that contains a relation symbol for $=$ has a quantified constraint satisfaction problem that is hard for L , since the problem $USTCON$ can be simulated by such a $QCSP$ (using two universal quantifiers).

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 $QCSP(\Gamma)$ is NP-complete.*

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

Theorem 7.2 (follows from [16]) *For any positive constraint language Γ , the problem $QCSP(\Gamma)$ is in NP.*

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

Proposition 7.3 *The problem $QCSP((D, T))$ is NP-hard.*

We now present a lemma that deals with positive languages preserved by an essential and surjective operation. The proof is based on known combinatorial results from universal algebra, and due to Michael Pinsker (and considerably simplifies the original proof of the authors).

Lemma 7.4 *Let Γ be a positive constraint language that is preserved by an essential operation on D with infinite image. Then Γ is preserved by all operations.*

In particular, such a constraint language is preserved by all binary injective operations.

Proposition 7.5 *Every positive relation R that is preserved by an injective binary operation g is negative.*

Proof. Let ϕ be a reduced formula that defines R (Definition 5.5). 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. \square

We can now prove Theorem 7.1.

Proof. (Theorem 7.1) The containment of $\text{QCSP}(\Gamma)$ in NP was discussed above. We prove NP-hardness of $\text{QCSP}(\Gamma)$ as follows. If all operations in $\text{sPol}(\Gamma)$ preserve T , then by Corollary 4.3 the NP-hard problem $\text{QCSP}((D, T))$ (Proposition 7.3) reduces to $\text{QCSP}(\Gamma)$. Otherwise, $\text{sPol}(\Gamma)$ contains an essential operation, and by Lemma 7.4 $\text{Pol}(\Gamma)$ contains all operations on D , and in particular $\text{Pol}(\Gamma)$ contains a binary injection. This can be used with Proposition 7.5 to show that Γ is negative. \square

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 express 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 PSPACE-hard.

Theorem 8.1 *Let Γ be a constraint language that is not positive. Then either Γ is negative or $[\Gamma]$ contains I .*

It will be helpful to consider the relation

$$S(x, y, z) \equiv (x = y \wedge y = z) \vee (x \neq y \wedge y \neq z \wedge x \neq z)$$

containing all tuples whose values are either all equal, or all different. Note that this relation consists of exactly two orbits of tuples. This will be relevant due to the following general lemma (which holds for arbitrary structures).

Lemma 8.2 *Let R be a k -ary relation that consists of m orbits of k -tuples. Then every operation f that violates R generates an 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, $l \leq m$. In case that $l = m$, we are done. In case that $l < m$, the result also follows, because we can then always obtain an m -ary operation that violates R by composing f' with projections. \square

Lemma 8.3 *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 . \square

It is clear that any operation that preserves \neq but violates S must be essential. The following lemma is one of the key results: we show that if a relation R is preserved by an operation that preserves \neq but violates S , then R must be negative.

Lemma 8.4 *Let R be a relation preserved by an operation f from $\text{Pol}(\{\neq\})$ that violates S . Then R is negative.*

Proof. (Theorem 8.1) Suppose that Γ is not positive, and $I \notin [\Gamma]$. We have to show that Γ is negative. By Lemma 8.3 and Theorem 4.2, $\text{sPol}(\Gamma)$ contains an operation f that violates S . Since Γ is not positive, Proposition 5.6 shows that f must preserve \neq . Lemma 8.4 then shows that Γ must be negative. \square

9 PSPACE-hardness

In this section, we prove that the QCSP over the relation I is PSPACE-hard.

Theorem 9.1 *$\text{QCSP}((D; I))$ is PSPACE-hard.*

Before giving the proof, we will need to introduce some notions and intermediate results.

Definition 9.2 Let us say that a relation $R \subseteq \{0, 1\}^n$ is force-definable if there exists a prenex formula

$$\Phi_{R, v_0, v_1}(b_0, b_1, x_1, \dots, x_n) = \mathcal{Q}\phi$$

over I such that:

- \mathcal{Q} is the quantifier prefix and ϕ is the quantifier-free part,
- the free variables of Φ_R are $\{b_0, b_1, x_1, \dots, x_n\}$,
- the quantifier prefix \mathcal{Q} contains v_0 and v_1 as its innermost two variables, and they are both existentially quantified,
- for any instantiation of the free variables $\{b_0, b_1, x_1, \dots, x_n\}$, the formula $\mathcal{Q}(\phi \wedge (v_0 = b_0) \wedge (v_1 = b_1))$ is true,
- when $\bar{t} \in \{0, 1\}^n$, if b_0 and b_1 are set arbitrarily and x_i is set to b_{t_i} (for all $i \in [n]$), then: the formula $\mathcal{Q}(\phi \wedge (v_0 \neq b_0 \vee v_1 \neq b_1))$ is false if and only if $\bar{t} \in R$
- (monotonicity) for any setting to the free variables $\{b_0, b_1, x_1, \dots, x_n\}$, if the formula $\mathcal{Q}(\phi \wedge (v_0 \neq b_0 \vee v_1 \neq b_1))$ is true, then changing the value of any variable x_i to a value not equal to b_0 nor b_1 preserves the truth of the given formula.

We call the formula $\Phi_{R, v_0, v_1}(b_0, b_1, x_1, \dots, x_n)$ the force-definition of R .

The intuition behind this definition is the following. Thinking of the prenex formula as a two-player game, when $\bar{t} \in \{0, 1\}^n$ is a tuple and the variables x_i are set according to \bar{t} (that is, by $x_i = b_{t_i}$), if $\bar{t} \in R$, then there is a way for the universal player to play so that v_0 is forced to be equal to b_0 , and v_1 is forced to be equal to b_1 ; and, if $\bar{t} \notin R$, then it is not possible for the universal player to force both of these equalities simultaneously.

Lemma 9.3 *There exists a polynomial-time algorithm that, given a boolean circuit C as input, produces a force-definition of the relation R_C containing, as tuples, the satisfying assignments of the circuit.*

Proof. It is known (see e.g. [10]) that from a boolean circuit C , it is possible to compute in polynomial time a primitive positive definition of R_C over the relations \mathcal{N} and \mathcal{P} defined as

$$\mathcal{N} = \{0, 1\}^3 \setminus \{(1, 1, 1)\}$$

$$\mathcal{P} = \{0, 1\}^3 \setminus \{(0, 0, 0)\}.$$

Note that \mathcal{N} contains all length 3 tuples with at least one coordinate having value 0 (the “negative” value), and \mathcal{P} contains all length 3 tuples with at least one coordinate having value 1 (the “positive” value).

Now, let $\exists s_1 \dots \exists s_k C$ be a primitive positive definition of $R_C(x_1, \dots, x_n)$, where C is the conjunction of atomic formulas $\mathcal{N}(u_1, u_2, u_3)$ and $\mathcal{P}(u_1, u_2, u_3)$ with $u_1, u_2, u_3 \in \{x_1, \dots, x_n, s_1, \dots, s_k\}$. Let us denote the atomic formulas $\mathcal{N}(u_1, u_2, u_3)$ in C by N_1, \dots, N_l and the atomic formulas $\mathcal{P}(u_1, u_2, u_3)$ in C by P_1, \dots, P_m . We now give a force-definition $\Phi_{R_C, v_0, v_1}(b_0, b_1, x'_1, \dots, x'_n)$ of R_C . Our force-definition has quantifier prefix:

$$\forall s'_1 \dots \forall s'_k \exists n_1 \dots \exists n_l \\ \exists t_1^n \dots \exists t_l^n \exists p_1 \dots \exists p_m \exists t_1^p \dots \exists t_m^p \exists v_0 \exists v_1$$

The quantifier-free part of our force-definition has the following constraints:

- For each constraint $N_i = \mathcal{N}(u_1, u_2, u_3)$ in C , there is a constraint $u_j = b_0 \rightarrow n_i = b_0$ for $j = 1, 2, 3$.

Similarly, for each constraint $P_i = \mathcal{P}(u_1, u_2, u_3)$ in C , there is a constraint $u_j = b_1 \rightarrow p_i = b_1$ for $j = 1, 2, 3$.

These constraints state that, for each constraint N_i (respectively, P_i), if it is true, then the value n_i must be set equal to b_0 (respectively, p_i must be set equal to b_1).

- There is a constraint $n_1 = b_0 \rightarrow t_1^n = b_0$ and, for each $i = 2, \dots, l$, a constraint $n_i = t_{i-1}^n \rightarrow t_i^n = t_{i-1}^n$.

Similarly, there is a constraint $p_1 = b_1 \rightarrow t_1^p = b_1$ and, for each $i = 2, \dots, m$, a constraint $p_i = t_{i-1}^p \rightarrow t_i^p = t_{i-1}^p$.

The first sequence of constraints ensure that if all of the values n_i are set equal to b_0 , then the variable t_1^n must also be set equal to b_0 ; also, if not all of the values n_i are set equal to b_0 , then the variable t_l^n may be set to any value.

Similarly, the second sequence of constraints ensure that if all of the values p_i are set equal to b_1 , then the variable t_m^p must also be set equal to b_1 ; and, if not all of the values p_i are set equal to b_1 , then the variable t_1^p may be set to any value.

- There are constraints $t_l^n = b_0 \rightarrow v_0 = b_0$ and $t_m^p = b_1 \rightarrow v_1 = b_1$.

We verify that this is indeed a force-definition, as follows. Consider an assignment to the variables $b_0, b_1, x'_1, \dots, x'_n$ where the x'_i represent a tuple $(x_1, \dots, x_n) \in \{0, 1\}^n$, that is, $x'_i = b_{x_i}$ for all $i \in [n]$.

First suppose that $(x_1, \dots, x_n) \in R_C$. Then, in the primitive positive definition of $R_C(x_1, \dots, x_n)$, there exists a boolean assignment to the variables s_1, \dots, s_k under which all constraints in C are true. Consider the assignment to the variables s'_i where $s'_i = b_{s_i}$ for all $i \in [k]$. We claim that any assignment to the remaining variables

of the force-definition (all of which are existentially quantified) that satisfies the quantifier-free part must set $v_0 = b_0$ and $v_1 = b_1$. Since every constraint N_i and P_i is satisfied by the assignment to $\{x_1, \dots, x_n, s_1, \dots, s_k\}$, by the first group of constraints in our force-definition, we have that all values n_i must be set equal to b_0 , and all values p_i must be set equal to b_1 . Then, by the second group of constraints, all values t_i^n must be set equal to b_0 , and all values t_i^p must be set equal to b_1 . By the last two constraints, we must have $v_0 = b_0$ and $v_1 = b_1$.

Next suppose that $(x_1, \dots, x_n) \notin R_C$. Then, for any setting to the variables s'_i , there exists some variable n_i that is not forced to b_0 , or some variable p_i that is not forced to b_1 . Let us suppose that n_j is not forced to b_0 (the other case is similar). In this case, it is possible to satisfy the second group of constraints by setting $t_1^n = \dots = t_{j-1}^n = b_0$, and the variables t_j^n, \dots, t_l^n to a fixed value outside of $\{b_0, b_1\}$. Then, v_0 can be set to any value. Note that all variables $\{p_1, \dots, p_m, t_1^p, \dots, t_m^p, v_1\}$ can be set equal to b_1 . \square

We now turn to the PSPACE-hardness proof. This proof takes inspiration from a PSPACE-hardness proof for the quantified constraint satisfaction problem for finite templates having a semilattice operation without unit [3].

Proof. (Theorem 9.1) We reduce from succinct graph unreachability. In this problem, the input is a boolean circuit with $2c$ inputs, which represents a graph G whose vertices are the tuples in $\{0, 1\}^c$; as well as two distinguished vertices $\bar{s}, \bar{t} \in \{0, 1\}^c$. There is a directed edge (\bar{x}, \bar{y}) in the graph if and only if the circuit returns true given input (\bar{x}, \bar{y}) . The question is to decide whether or not there is a directed path in the graph from \bar{s} to \bar{t} . This problem is known to be PSPACE-complete [18].

Define $R_i \subseteq \{0, 1\}^{2c}$ to be the relation containing all tuples (\bar{x}, \bar{y}) such that there exists a directed path in G from \bar{x} to \bar{y} of length less than or equal to 2^i . We claim that a force-representation of R_c can be computed from the input circuit in polynomial time. We give a force-representation for each R_i from $i = 0, \dots, c$, using a recursive construction; from the construction, it will be clear that R_c can be constructed in polynomial time.

Representations of the R_i . To obtain a force-representation for R_0 , we simply take the input circuit, augment it so that it always returns true on inputs (\bar{x}, \bar{x}) , and appeal to Lemma 9.3.

Now, we assume that we have a force-representation for R_i , and give a force-representation for R_{i+1} . Let $\Phi_{R_i, z_0, z_1}(b_0, b_1, (\bar{u}, \bar{v})) = \mathcal{Q}_i \phi_i$ be a force-representation for R_i with the indicated variables. Let E denote the relation $\{(\bar{x}, \bar{y}) \in \{0, 1\}^{4c} : \forall i, x_i = y_i\}$. By Lemma 9.3, the boolean relation E has force-representations

$$\Phi_{E, v'_0, v'_1}(z_0, z_1, ((\bar{u}, \bar{v}), (\bar{x}, \bar{q}))) = Q' \phi'$$

and

$$\Phi_{E, v''_0, v''_1}(z_0, z_1, ((\bar{u}, \bar{v}), (\bar{q}, \bar{y}))) = Q'' \phi''$$

with the indicated variables. The following is a force-representation for R_{i+1} ; note that $\bar{x}, \bar{y}, \bar{q}, \bar{u}, \bar{v}$ all denote vectors of length c .

$$\begin{aligned} & \Phi_{R_{i+1}, w_0, w_1}(b_0, b_1, (\bar{x}, \bar{y})) = \\ & \forall \bar{q} Q' Q'' \forall \bar{u} \forall \bar{v} Q_i \exists w_0 \exists w_1 \phi' \wedge \phi'' \wedge \phi_i \\ & \wedge (v'_0 = v''_0 \rightarrow w_0 = v''_0) \wedge (v'_1 = v''_1 \rightarrow w_1 = v''_1) \end{aligned}$$

First, suppose that there is a path from \bar{x} to \bar{y} of length less than or equal to 2^i , that is, $(\bar{x}, \bar{y}) \in R_i$. (Here, we associate the boolean values 0, 1 with the values b_0, b_1 .) Then, there exists a tuple \bar{q} such that $(\bar{x}, \bar{q}), (\bar{q}, \bar{y}) \in R_{i-1}$. We want to show that the universal player can play so that the existential player is forced to set $w_0 = b_0$ and $w_1 = b_1$.

Let the universal player play inside Q' as if $(\bar{u}, \bar{v}) = (\bar{x}, \bar{q})$, so as to force $v'_0 = z_0$ and $v'_1 = z_1$. Observe that if the existential player does not set $v'_0 = z_0$ and $v'_1 = z_1$, then the universal player can falsify a constraint by subsequently setting $(\bar{u}, \bar{v}) = (\bar{x}, \bar{q})$. Similarly, the universal player can play inside Q'' in a way that (s)he forces $v''_0 = z_0$ and $v''_1 = z_1$.

We claim that, in fact, at this point in the game (after the variables in Q' and Q'' have been set), it must be the case that v'_0 and v''_0 must both be set to b_0 , and v'_1 and v''_1 must both be set to b_1 . If we do not have (for example) $v'_0 = b_0$ and $v'_1 = b_1$, then the universal player can set $(\bar{u}, \bar{v}) = (\bar{x}, \bar{q})$, and then, to satisfy the constraints of ϕ_i , the existential player must set $z_0 = b_0$ and $z_1 = b_1$; since we already established that the existential player must obey $v'_0 = z_0$ and $v'_1 = z_1$, it follows that some constraint is falsified.

Finally, observe that to satisfy the constraints $(v'_0 = v''_0 \rightarrow w_0 = v''_0) \wedge (v'_1 = v''_1 \rightarrow w_1 = v''_1)$, the existential player must set $w_0 = b_0$ and $w_1 = b_1$.

The reduction. Having computed a force-representation

$$\Phi_{R_c, w_0, w_1}(b_0, b_1, (\bar{x}, \bar{y})) = \mathcal{Q}_c \phi_c$$

of R_c , we now show how to use it to give an instance of QCSP($(D; I)$) that is true if and only if there is no path from \bar{s} to \bar{t} in the succinctly represented graph. The instance created is

$$\begin{aligned} & \exists b_0 \exists b_1 \exists x \exists y \mathcal{Q}_c (\phi_c \wedge (b_0 \neq b_1) \wedge (\bar{x} = \bar{s}) \\ & \wedge (\bar{y} = \bar{t}) \wedge (v_0 \neq b_0 \vee v_1 \neq b_1)). \end{aligned}$$

Note that a disjunction $a \neq b \vee c \neq d$, which is used in this expression, has the primitive positive definition $\exists b' \exists d' \exists t (I(a, b, b') \wedge I(b, b', t) \wedge I(c, d, d') \wedge I(d, d', t) \wedge (b' \neq d'))$, and $x \neq y$ has the $\forall \exists \wedge$ -definition $\forall z. I(x, y, z)$; therefore, these expressions can be used in our reduction (Lemma 3.1). \square

10 Concluding Remarks

We would like to remark that the classification of the computational complexity of $\text{QCSP}(\Gamma)$ 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 PSPACE-complete or NP-complete. This follows from the fact that all classes of equality constraint languages considered in this paper are defined in terms of *preservation* under certain polymorphisms that can be tested algorithmically; this will be shown in the full version of the paper.

Acknowledgements We thank Jerome Keisler for helpful explanations concerning [15], and Michael Pinsker for friendly permission to include some results from forthcoming joint work [1].

References

- [1] M. Bodirsky, H. Chen, and M. Pinsker. The locally closed clones above the permutations. In preparation.
- [2] M. Bodirsky and J. Kára. The complexity of equality constraint languages. *Theory of Computing Systems, to appear; an extended abstract appeared in the Proceedings of the International Computer Science Symposium in Russia (CSR'06)*, 2007.
- [3] F. Boerner, A. Bulatov, H. Chen, P. Jeavons, and A. Krokhin. The complexity of constraint satisfaction games and qcsp. *Preprint N106023, Isaac Newton Institute, Cambridge. Submitted for journal publication*, 2006.
- [4] 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.
- [5] A. Bulatov. Tractable conservative constraint satisfaction problems. In *Proceedings of LICS'03*, pages 321–330, 2003.
- [6] A. Bulatov and V. Dalmau. A simple algorithm for Mal'tsev constraints. *SIAM J. Comp.*, 36(1):16–27, 2006.
- [7] A. Bulatov, A. Krokhin, and P. G. Jeavons. Classifying the complexity of constraints using finite algebras. *SIAM Journal on Computing*, 34:720–742, 2005.
- [8] P. J. Cameron. *Oligomorphic Permutation Groups*. Cambridge University Press, 1990.
- [9] Chang and Keisler. *Model theory*. Princeton University Press, 1977.
- [10] N. Creignou, S. Khanna, and M. Sudan. *Complexity Classification of Boolean Constraint Satisfaction Problems*. SIAM Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, 2001.
- [11] M. V. Emil W. Kiss. On tractability and congruence distributivity. In *Proceedings of LICS'06*, pages 221–230, 2006.
- [12] P. Hell and J. Nešetřil. *Graphs and Homomorphisms*. Oxford University Press, 2004.
- [13] W. Hodges. *A shorter model theory*. Cambridge University Press, 1997.
- [14] P. Jeavons, D. Cohen, and M. Gyssens. Closure properties of constraints. *Journal of the ACM*, 44(4):527–548, 1997.
- [15] J. Keisler. Reduced products and Horn classes. *Trans. Amer. Math. Soc.*, 117:307–328, 1965.
- [16] D. Kozen. Positive first-order logic is NP-complete. *IBM Journal of Research and Development*, 25(4):327–332, 1981.
- [17] D. Marker. *Model Theory: An Introduction*. Springer, 2002.
- [18] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.
- [19] O. Reingold. Undirected st-connectivity in log-space. *Electronic Colloquium on Computational Complexity (ECCC)*, 094, 2004.
- [20] T. J. Schaeffer. The complexity of satisfiability problems. In *Proceedings of STOC'78*, pages 216–226, 1978.
- [21] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time: Preliminary report. In *STOC*, pages 1–9, 1973.
- [22] A. Szendrei. *Clones in universal Algebra*. Seminaire de mathematiques superieures. Les Presses de L'Universite de Montreal, 1986.