The Complexity of Equality Constraint Languages

Manuel Bodirsky[1] and Jan Kára[2]

[1] Algorithms and Complexity Department, Humboldt University, Berlin, bodirsky@informatik.hu-berlin.de

[2] Department of Applied Mathematics, Charles University, Prague, kara@kam.mff.cuni.cz **

Abstract. We apply the algebraic approach to infinite-valued constraint satisfaction to classify the computational complexity of all constraint satisfaction problems with templates that have a highly transitive automorphism group. A relational structure has such an automorphism group if and only if all the constraint types are Boolean combinations of the equality relation, and we call the corresponding constraint languages *equality constraint languages*. We show that an equality constraint language is tractable if it admits a constant unary or an injective binary polymorphism, and is NP-complete otherwise.

Keywords: Constraint Satisfaction, Logic in Computer Science, Computational Complexity, Clones on Infinite Domains

1 Introduction

In a constraint satisfaction problem we are given a set of variables and a set of constraints on those variables, and want to find an assignment of values to the variables such that all the constraints are satisfied. The computational complexity of the constraint satisfaction problem depends on the constraint language that we are allowed to use in the instances of the constraint satisfaction problem, and attracted a lot of interest in recent years; see e.g. [6] for an introduction to the state-of-the-art of the techniques used to study the computational complexity of constraint satisfaction problems.

Formally, we can define constraint satisfaction problems (CSPs) as *homomorphism problems* for relational structures. Let Γ be a (not necessarily finite) structure with a relational signature τ. Then the constraint satisfaction problem $\mathrm{CSP}(\Gamma)$ is a computational problem where we are given a *finite* τ-structure S and want to know whether there is a homomorphism from S to Γ; for the detailed definitions, see Section 2. We show two examples.

Example 1. Let Γ be the relational structure $(\mathbb{N}; =, \neq)$. Then $\mathrm{CSP}(\Gamma)$ is the computational problem to determine for a given set of equality or inequality constraints on a finite set of variables whether the variables can be mapped to the natural numbers such that variables x, y with a constraint $x = y$ are mapped to the same value and variables x, y with a constraint $x \neq y$ are mapped to distinct values.

** The second author has been supported by a Marie Curie fellowship of the graduate program "Combinatorics, Geometry, and Computation", HPMT-CT-2001-00282. Also supported by project 1M0021620808 of the Ministry of Education of the Czech Republic.

This problem is tractable: for this, we consider the undirected graph on the variables of an instance S of CSP(Γ), where two variables x and y are joined iff there is a constraint $x = y$ in S. Then it is easy to see that S does not have a solution if and only if it contains an inequality-constraint $x \neq y$ such that y is reachable from x in the graph defined above. Clearly, such a reachability test can be performed in polynomial time.

Example 2. Let Γ be the relational structure $(\mathbb{N}; S)$, where S is the ternary relation $S := \{ (x_1, x_2, x_3) \in \mathbb{N}^3 \mid (x_1 = x_2 \wedge x_2 \neq x_3) \vee (x_1 \neq x_2 \wedge x_2 = x_3) \}$. Here the problem CSP($\Gamma$) turns out to be NP-complete (see Section 5).

In this paper we consider constraint satisfaction problems where the infinite template $\Gamma = (D; R_1, \ldots, R_k)$ has a *highly transitive* automorphism group, i.e., if every permutation of D is an automorphism of Γ. That is, we study the constraint satisfaction problems for templates with the highest possible degree of symmetry. We will see in Section 2 that Γ has a highly transitive automorphism group if and only if all relations $R_1, \ldots, R_k$ can be defined with a Boolean combination of atoms of the form $x = y$. (A Boolean combination is a formula built from atomic formulas with the usual connectives of conjunction, disjunction, and negation.) We say that such a relational structure defines an *equality constraint language*. Later, we also discuss the case where the template has infinitely many relation symbols $R_1, R_2, \ldots$ Note that Example 1 and 2 are both equality constraint languages.

The main result of this paper is a full classification of the computational complexity of equality constraint languages. They are either tractable, or NP-complete. The containment in NP is easy to see: a nondeterministic algorithm can guess which variables in an instance S denote the same element in Γ and can verify whether this gives rise to a solution for S. To prove that certain equality constraint languages are NP-hard (Section 5) we apply the algebraic approach to constraint satisfaction, which was previously mainly applied to constraint satisfaction with finite templates.

Some equality constraint languages are tractable. These languages are described by certain closure properties. The most interesting languages here are those that are *closed under an injective binary operation*. The polynomial-time algorithm for such languages, which is presented in Section 6, is an instantiation of the relational consistency algorithm as introduced in [8]. Our contribution here is the proof that this algorithm is *complete* for equality constraint languages that are closed under an injective binary polymorphism, this is, the algorithm rejects an instance if and only if the instance does not have a solution.

2 Fundamental Concepts for the Algebraic Approach

We introduce classical concepts that are fundamental for the algebraic approach to constraint satisfaction. A general introduction to these concepts is [12]; for clones and polymorphisms we refer to [17].

Structures. A *relational language* τ is a (here always at most countable) set of *relation symbols* R_i, each associated with a finite *arity* k_i. A *(relational) structure* Γ over the *(relational) language* τ (also called τ-*structure*) is a countable 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. For a τ-structure Γ and $R \in \tau$ it will also be convenient to say that $R(u_1, \ldots, u_k)$ *holds in* Γ iff $(u_1, \ldots, u_k) \in R$. We sometimes write $\overline{u}$ for a tuple $(u_1, \ldots, u_k)$ of some length k. If we add relations to a given structure Γ, we call the resulting structure Γ' an *expansion* of Γ, and Γ is called a *reduct* of Γ'.

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 $(a_1, \ldots, a_n)$, if $(a_1, \ldots, a_n) \in R^\Gamma$, then $(f(a_1), \ldots, f(a_n)) \in R^{\Gamma'}$. In this case we say that the map f *preserves* the relation R. Isomorphisms from Γ to Γ are called *automorphisms*, and homomorphisms from Γ to Γ are called *endomorphisms*. The set of all automorphisms of a structure Γ is a group, and the set of all endomorphisms of a structure Γ is a monoid with respect to composition.

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 $R(x_1^i, \ldots, x_m^i)$ holds in Γ for all $1 \leq i \leq k$, then $R\big(f(x_1^1, \ldots, x_1^k), \ldots, f(x_m^1, \ldots, x_m^k)\big)$ holds in Γ. If f preserves all relations of a relational τ-structure Γ, we say that f is a polymorphism of Γ. In other words, f is a homomorphism from $\Gamma^k = \Gamma \times \ldots \times \Gamma$ to Γ, where $\Gamma_1 \times \Gamma_2$ is the *(categorical- or cross-) product* of the two relational τ-structures Γ_1 and Γ_2. Hence, the unary polymorphisms of Γ are the endomorphisms of Γ, and the unary bijective polymorphisms are the automorphisms 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

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

A *clone* F is a set of operations from O that is closed under compositions and that contains all projections. We write D_F for the *domain* D of the clone F. It is easy to verify that the set $Pol(\Gamma)$ of all polymorphisms of Γ is a clone with the domain D_Γ. Moreover, $Pol(\Gamma)$ is also closed under interpolations: we say that an operation $f \in O$ is *interpolated* by a set $F \subseteq O$ if for every finite subset B of D there is some operation $g \in F$ such that $f|_B = g|_B$ (f restricted to B equals g restricted to B, i.e., $f(\overline{s}) = g(\overline{s})$ for every $\overline{s} \in B^k$). 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*. The following is a well-known fact:

Proposition 1 (see e.g. [16]). *A set $F \subseteq O$ of operations is locally closed if and only if F is the set of polymorphisms of Γ for some relational structure Γ.*

An operation is called *essentially unary* iff 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 iff there is no $k{-}1$-ary operation f' such that $f(x_1, \ldots, x_k) = f'(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k)$. Hence, an essentially unary operation is an operation that depends on one argument only. 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 $f(x_1, \ldots, x_k) \neq f(x_1, \ldots, x_{i-1}, x_i', x_{i+1}, \ldots, x_k)$. We refer to [16] and [17] for a general introduction to clones.

3 The Algebraic Approach

A τ-formula is called *primitive positive*, if it has the form $\exists x_1 \dots x_k . \psi_1 \wedge \dots \wedge \psi_l$, where ψ_i is an atomic τ-formula that might contain free variables and existentially quantified variables from $x_1, \dots, x_k$. The atomic formula ψ_i might also be of the form $x = y$. A formula is called *existential positive*, if it is a disjunctive combination of primitive positive formulas (equivalently, if it is a first-order formula without universal quantifiers and negations). Every formula with k free variables defines on a structure Γ a k-ary relation. Primitive positive definability of relations is an important concept in constraint satisfaction because primitive positive definable relations can be 'simulated' by the constraint satisfaction problem. The following is frequently used in hardness proofs for constraint satisfaction problems; see e.g. [13].

Lemma 1. *Let Γ be a relational structure and let R be a relation that has a primitive positive definition in Γ. Then the constraint satisfaction problems of Γ and of the expansion of Γ by R have the same computational complexity.*

The algebraic approach to constraint satisfaction (see e.g. [4, 5, 13]) is based on the following preservation statements that characterize syntactic restrictions of first-order definability.

Theorem 1 (from [3, 10, 14]). *Let Γ be a finite relational structure. Then*

1. *A relation R has a first-order definition in Γ if and only if it is preserved by all automorphisms of Γ;*
2. *A relation R has an existential positive definition in Γ if and only if it is preserved by all endomorphisms of Γ;*
3. *A relation R has a primitive positive definition in Γ if and only if it is preserved by all polymorphisms of Γ.*

These statements do not hold for infinite structures in general. However, we have the following.

Theorem 2 (from [1,2]). *Let Γ be a countably infinite relational structure. Then Statement 1 of Theorem 1 holds if and only if Γ is* ω-categorical*, i.e., if the first-order theory of Γ has only one countable model up to isomorphism. For ω-categorical Γ, Statements 2 and 3 hold as well.*

Let G be a permutation group on a countable infinite set D. An *orbit of k-tuples in Γ* is a largest set O of k-tuples in Γ such that for all $\overline{s}, \overline{t} \in O$ there is a permutation α of Γ such that $(\alpha(s_1), \dots, \alpha(s_k)) = (t_1, \dots, t_k)$. A permutation group G on a countably infinite set D is called *oligomorphic*, if it has only finitely many orbits of k-tuples from D, for all $k \geq 1$; see [7]. The next theorem can be seen as a reformulation of the theorem of Ryll-Nardzewski, Engeler, and Svenonius (see [12]), and is also closely related to the first part of Theorem 2.

Theorem 3 (See [7]). *Let Γ be a relational structure. Then the following are equivalent.*

– *Γ is ω-categorical;*
– *the automorphism group of Γ is oligomorphic;*

– *every k-ary first-order definable relation in Γ is the union of a finite number of orbits of k-tuples of the automorphism group of Γ.*

Now it is easy to see that a relational structure $\Gamma = (V; R_1, R_2, \dots)$ has a highly transitive automorphism group if and only if all relations can be defined with Boolean combinations of the equality relation. Clearly, such relations are preserved by all permutations of V. On the other hand, if Γ has a highly transitive automorphism group, it is in particular ω-categorical. Hence, every k-ary relation R from Γ is the union of a finite number of orbits of k-tuples of the automorphism group of Γ. It is easy to see that the orbits of k-tuples of a highly transitive permutation group can be described by a conjunction of equality and inequality relations.

4 Representations of Relations

From now on, unless stated otherwise, $\Gamma = (D; R_1, R_2, \dots)$ is a relational structure on a countably infinite domain D where every relation R_i can be defined by a Boolean combination of atoms of the form $x = y$. Note that the automorphism group of Γ is the full symmetric group on D, which is clearly oligomorphic.

Both the hardness results and the algorithm for equality constraint languages use a special representation of the relations in Γ, which we are now going to describe. Theorem 3 implies that every k-ary relation in Γ is a union of orbits of k-tuples of the automorphism group of Γ. Let $\overline{s}$ be a k-tuple from one of these orbits. We define the equivalence relation ρ on the set $\{1, \dots, k\}$ that contains those pairs $\{i, j\}$ where $s_i = s_j$. Clearly, all tuples in the orbit lead to the same equivalence relation ρ. Hence, every k-ary relation R in Γ corresponds uniquely to a set of equivalence relations on $\{1, \dots, k\}$, which we call the *representation* of R. Sometimes we identify a relation R from Γ with its representation and for example freely write $\rho \in R$ if ρ is an equivalence relation from the representation of R. Let $|R|$ denote the number of orbits of k-tuples contained in R. Hence, $|R|$ also denotes the number of equivalence relations in the representation of R.

Definition 1. *Let ρ and ρ' be equivalence relations on a set X. We say that ρ is* finer *than ρ', and write $\rho \subseteq \rho'$, if $\rho(x, y)$ implies $\rho'(x, y)$ for each $x, y \in X$. We also say that in this case ρ' is* coarser *than ρ. The* intersection *of two equivalence relations ρ and ρ', denoted by $\rho \cap \rho'$, is the equivalence relation σ such that $\sigma(x, y)$ if and only if $\rho(x, y)$ and $\rho'(x, y)$. Finally, let $c(\rho)$ denote the number of equivalence classes in ρ.*

Lemma 2. *For a k-ary relation R in an equality constraint language on a countable set D the following are equivalent.*

1. *R is preserved by every injection of D^2 into D;*
2. *R is preserved by an injective binary operation on D;*
3. *R is preserved by a binary operation f and there are two k-element subsets S_1, S_2 of the domain such that f restricted to $S_1 \times S_2$ is injective;*
4. *The representation of R is* closed under intersections, *i.e., $\rho \cap \rho' \in R$ for all equivalence relations $\rho, \rho' \in R$;*

Proof. The implication from (1) to (2) and from (2) to (3) is immediate. Let ρ and ρ' be two equivalence relations from the representation of R. Pick two k-tuples $\overline{s}$ and $\overline{s}'$ in R

that lie in the orbits that are described by ρ and ρ'. Now, let f be a binary operation of D that is injective on its restriction to $S_1 \times S_2$ for two k-element subsets S_1, S_2. Let α_1 and α_2 be permutations of D that map the entries of the k-tuples $\overline{s}$ and $\overline{s}'$ to S_1 and S_2, respectively. Then by injectivity of f the k-tuple $\overline{s}'' := (f(\alpha_1(s_1), \alpha_2(s'_1)), \ldots, f(\alpha_1(s_k), \alpha_2(s'_k)))$ satisfies $s''_i = s''_j$ if and only if $\rho(i,j)$ and $\rho'(i,j)$. Hence, we found a tuple in R that lies in the orbit that is described by $\rho \cap \rho'$, which is therefore also contained in the representation of R, and therefore (3) implies (4). Every injection of D^2 into D preserves every relation with an intersection-closed representation, because it maps two tuples that correspond to equivalence relations ρ and ρ' to a tuple that corresponds to $\rho \cap \rho'$. We thus proved that (4) implies (1). $\square$

If a relation R has a representation that is closed under intersections, we also write that R is *$\cap$-closed*. The proofs can be found in the appendix.

Corollary 1. *An operation f on a countable set D and the permutations on D locally generate an injective binary operation g if and only if every equality constraint relation that is preserved by f is $\cap$-closed.*

Proof. If f and the permutations locally generate an injective binary operation g, then every relation R that is preserved by f is also preserved by g, and Lemma 2 shows that R is $\cap$-closed. Conversely, if every equality constraint relation R preserved by f is $\cap$-closed, we claim that f and the permutations locally generate all injective binary operations. Suppose the contrary. Then there is a relation R that is preserved by f but not by an injective binary operation g. An application of Lemma 2 in the other direction shows that R cannot be $\cap$-closed, contradicting the assumption. $\square$

5 A Generic Hardness Proof

In this section we prove that every equality constraint language without a constant unary or an injective binary polymorphism is NP-hard. Let us start with a fundamental lemma on non-injective endomorphisms.

Lemma 3. *If Γ has a non-injective endomorphism f, then Γ also has a constant endomorphism.*

Proof. Let f be an endomorphism of Γ such that $f(x) = f(y)$ for two distinct points x, y from D. Let $a_1, a_2, \ldots$ be an enumeration of D. We construct an infinite sequence of endomorphisms $e_1, e_2, ...$ where e_i is an endomorphism that maps the points $a_1, \ldots, a_i$ to a_1. This suffices, since by local closure the mapping defined by $e(x) = a_1$ for all x is an endomorphism of Γ.

For e_1 we take the identity map, which clearly is an endomorphism with the desired properties. To define e_i for $i \geq 2$ let α be an automorphism of Γ that maps $a_1 = e_{i-1}(a_1) = \cdots = e_{i-1}(a_{i-1})$ to x, and $e_{i-1}(a_i)$ to y. Then the endomorphism $f(\alpha(e_{i-1}))$ is constant on $a_1, \ldots, a_i$. There is also an automorphism α' that maps $f(\alpha(e_{i-1}(a_1)))$ to a_1. Then $e_i := \alpha'(f(\alpha(e_{i-1})))$ is an endomorphism with the desired properties. $\square$

Lemma 4. *If Γ does not have a constant endomorphism, then there is a primitive positive definition of the relation $x \neq y$ in Γ.*

Proof. Suppose Γ has a k-ary polymorphism f that does not preserve $\neq$, i.e., there are k-tuples $\overline{u}$ and $\overline{v}$ such that $u_i \neq v_i$ for all $i \in \{1, \ldots, k\}$, but $f(\overline{u}) \neq f(\overline{v})$. Let $\alpha_2, \ldots, \alpha_k$ be permutations of D that map u_1 to u_i and v_1 to v_i. Then the endomorphism $g(x) := f(x, \alpha_2(x), \ldots, \alpha_k(x))$ is not injective, because $g(u_1) = f(u_1, \ldots, u_k) = f(v_1, \ldots, v_k) = g(v_1)$, and by Lemma 3 locally generates a constant, in contradiction to the assumptions. Hence, every polymorphism of Γ preserves $\neq$, and by Theorem 2 the relation $\neq$ has a primitive positive definition. $\square$

Due to the following lemma we can focus on binary operations in some later proofs.

Lemma 5. *Every essentially at least binary operation together with all permutations locally generates a binary operation that depends on both arguments.*

Proof. Let k be a k-ary operation, where $k > 2$, that depends on all arguments. In particular, f depends on the first argument, and hence there are two k-tuples $(a_1, \ldots, a_k)$ and $(a_1', a_2, \ldots, a_k)$ with $f(a_1, \ldots, a_k) \neq f(a_1', a_2, \ldots, a_k)$. Suppose first that there are $b_1, \ldots, b_k$ such that $b_i \neq a_i$ for $i \geq 2$ and $f(b_1, b_2, \ldots, b_k) \neq f(b_1, a_2, \ldots, a_k)$. We can then define permutations α_i of D for $i \geq 3$, such that a_2 is sent to a_i and d_2 is sent to d_i. The binary operation g defined by $g(x, y) = f(x, y, \alpha_3(y), \ldots, \alpha_k(y))$ depends on both arguments, as $g(a_1, a_2) \neq g(a_1', a_2)$ and $g(b_1, b_2) \neq g(b_1, a_2)$, and hence we are done in this case.

So suppose that for every b_1 and every $b_2, \ldots, b_k$ such that $b_i \neq a_i$ for $i \in \{2, \ldots, k\}$ it holds that $f(b_1, b_2, \ldots, b_k) = f(b_1, a_2, \ldots, a_k)$. Since f depends on the second coordinate, there are elements $c_1, c_2, \ldots, c_k$ and c_2' with $f(c_1, \ldots, c_k) \neq f(c_1, c_2', c_3, \ldots, c_k)$. The value $f(c_1, a_2, \ldots, a_k)$ can be equal to either $f(c_1, \ldots, c_k)$ or to $f(c_1, c_2', c_3, \ldots, c_k)$, but not to both. We can assume without loss of generality that $f(c_1, \ldots, c_k) \neq f(c_1, a_2, \ldots, a_k)$. Let us choose $d_2, \ldots, d_k$ such that $d_i \neq a_i$ and $d_i \neq c_i$ for $i \in \{2, \ldots, k\}$. Since c_i and d_i are distinct for all $2 \leq i \leq k$, we can define permutations α_i of D for $i \geq 3$ such that b_2 is sent to b_i and c_2 is sent to c_i.

We claim that the operation g defined by $g(x, y) := f(x, y, \alpha_3(y), \ldots, \alpha_k(y))$ depends on both arguments. Indeed, from the beginning of the previous paragraphs we know that $g(a_1, d_2) = f(a_1, d_2, \ldots, d_k) = f(a_1, a_2, \ldots, a_k)$, and that $g(a_1', d_2) = f(a_1', d_2, \ldots, d_k) = f(a_1', a_2, \ldots, a_k)$. By the choice of the values $a_1, \ldots, a_k$ and a_1' these two values are distinct, and we have shown that g depends on the first argument. For the second argument, note that $g(c_1, d_2) = f(c_1, d_2, \ldots, d_k) = f(c_1, a_2, \ldots, a_k)$ and that $g(c_1, c_2) = f(c_1, c_2, \ldots, c_k)$. But in the previous paragraph we also saw that these two values are distinct, and hence g also depends on the second argument. $\square$

Now comes the central argument.

Theorem 4. *Let f be a binary operation that depends on both arguments. Then f together with all permutations locally generates either a constant unary operation or a binary injective operation.*

Proof. Suppose that f does not locally generate a constant operation. We want to use Corollary 1 and show that every equality constraint relation R that is preserved by f is $\cap$-closed, which implies that f locally generates a binary injective polymorphism. Suppose for contradiction that R is an n-ary equality constraint relation, $n \geq 2$, that is closed under f but not $\cap$-closed, i.e., there are two equivalence relations ρ and ρ' in R such that

$\rho \cap \rho'$ is not in R. Choose ρ and ρ' such that $(c(\rho), c(\rho'))$ is lexicographically maximal. Let $\overline{s} := (s_1, \ldots, s_n)$ and $\overline{t} := (t_1, \ldots, t_n)$ be n-tuples of D that have the equivalence relations ρ and ρ'. Because ρ is not finer than ρ' we can find indices p and q such that $s_p = s_q$, $t_p \neq t_q$. Let r be the number of equivalence classes of ρ that are contained in the equivalence class of p in ρ'. Choose p and q such that r is minimal.

Consider $2n-1$ distinct elements $a_1, \ldots, a_{2n-1}$ from D. By the infinite pigeon-hole principle, there is an infinite subset S_1 of D such that $f(a_1, b) = f(a_1, b')$ for all $b, b' \in S_1$, or $f(a_1, b) \neq f(a_1, b')$ for all $b, b' \in S_1$. We apply the same argument to a_2 instead of a_1, and S_1 instead of D, and obtain an infinite subset S_2 of S_1. The argument can be iterated to obtain an infinite subset S_{2n-1} such that for all $a \in \{a_1, \ldots, a_{2n-1}\}$ we either have $f(a, b) \neq f(a, b')$ for all $b, b' \in B$, or $f(a, b) = f(a, b')$ for all $b, b' \in B$. Then there is also an n-element subset A of $\{a_1, \ldots, a_{2n-1}\}$ and an n-element subset B of S_{2n-1} such that either $f(a, b) \neq f(a, b')$ for all $a \in A$ and $b, b' \in B$, or $f(a, b) = f(a, b')$ for all $a \in A$ and $b, b' \in B$. Not-e that in the latter case $f(a, b) \neq f(a', b)$ for all distinct elements $a, a' \in A$, and $b \in B$. Otherwise, if $f(a, b) = f(a', b)$, then f does not preserve the inequality relation, because there is a $b' \in B$ such that $b' \neq b$ and $f(a, b) = f(a, b')$, and hence $f(a, b) = f(a', b')$, but $a \neq a'$ and $b \neq b'$. But this is impossible, because Lemma 3 shows that in this case f locally generates a constant operation. Therefore, we found two n-element sets A and B such that either $f(a, b) \neq f(a', b)$ for all $a, a' \in A$ and $b \in B$, or $f(a, b) \neq f(a, b')$ for all $a \in A$ and $b, b' \in B$. Without loss of generality we assume that the first case applies.

Since f cannot only depend on the first argument, there are elements u, v_1, and v_2 in D such that $v_1 \neq v_2$ and $f(u, v_1) \neq f(u, v_2)$. We can assume that v_2 is from B: For this, consider any element v' of B. If $f(u, v') \neq f(u, v_1)$, we choose v' instead of v_2 and are done. If $f(u, v') = f(u, v_1)$, then $f(u, v') \neq f(u, v_2)$, and we choose v' instead of v_2 and v_2 instead of v_1. We can also assume that $f(u, v) \neq f(u', v)$ for all $u' \in A$, $v \in B$: The reason here is that if there are elements $a \in A$ and $b_1, b_2 \in B$ such that $f(a, b_1) \neq f(a, b_2)$ we choose $u = a$, $v_1 = b_1$, and $v_2 = b_2$. Otherwise, we know that $f(a, b_1) = f(a, b_2)$ for all $a \in A$ and $b_1, b_2 \in B$. But then, $f(u, v) = f(u', v)$ is impossible for all $u' \in A$ and $v \in B$ due to Lemma 3.

Let α_1 be a permutation of D that maps $s_p = s_q$ to u and the other entries in $\overline{s}$ to A. Let α_2 be a permutation of D that maps t_p to v_1, t_q to v_2, and the other entries in $\overline{t}$ to B. Consider the equivalence relation σ of the tuple $(f(\alpha_1(s_1), \alpha_2(t_1)), \ldots, f(\alpha_1(s_n), \alpha_2(t_n)))$. Because f preserves R, we know that σ is contained in R. If $r = 0$, then due to the way we apply the operation f to $\alpha_1(\overline{s})$ and $\alpha_2(\overline{t})$ it is easy to see that σ has more equivalence classes than ρ, contradicting the maximal choice of ρ. If $r \geq 1$, then σ has more equivalence classes than ρ', for the following reason. Every equivalence class C of ρ' either consists of a union of equivalence classes from ρ, or contains an element from an equivalence class in ρ that is not contained in C. But also in the latter case, by the choice of p and q such that r is minimal, we can infer that C contains some equivalence class from ρ. Hence, in both cases we can associate in that way one equivalence class from ρ to every class in ρ'. Due to the way we apply the operation f to $\alpha_1(\overline{s})$ and $\alpha_2(\overline{t})$, all these equivalence classes correspond to distinct equivalence classes in σ. Moreover, $f(\alpha_1(s_q), \alpha_2(t_q))$ will lie in yet another equivalence class of σ. Thus, σ has more equivalence classes than ρ'. Since σ is not coarser than ρ, the existence of the relations ρ and σ then contradicts the choice of ρ and ρ' where $(c(\rho), c(\rho'))$ was lexicographically maximal. $\square$

Hence, if the template is not preserved by a constant unary or an injective binary operation, we have a primitive positive definition of every first-order definable relation, in particular for the relation S that was defined in Example 2 in the introduction.

Lemma 6. *If the relation S has a primitive positive definition in Γ, then CSP(Γ) is NP-hard.*

Proof. First observe that by identification of arguments x and y, if S has a primitive positive definition in Γ, then the inequality relation has a primitive positive definition in Γ as well. We prove the NP-hardness by reduction from the NP-hard problem 3-COLORING [9]. Let $G = (V, E)$ be a graph that is an instance of 3-COLORING. We construct an instance of CSP(Γ) that has a polynomial size in $|V|$ and $|E|$ and is satisfiable if and only if G has a proper 3-coloring. Lemma 1 asserts we can use inequality constraints and the relation S to formulate this instance. The set of variables in this instance is $V \cup V' \cup \{c_1, c_2, c_3\}$, where V' is a copy of V, and c_1, c_2, c_3 are three new variables representing colors. We impose inequality constraints on each pair in c_1, c_2, c_3 and on each pair (u, v) for $uv \in E$. We impose the constraint S on (c_1, v', c_2) for each $v' \in V'$, and on (v', v, c_3) for each $v \in V$ where v' is the copy of v in V'. By construction, a solution to these constraints induces a proper 3-coloring of G. Conversely, a simple case analysis shows that any proper 3-coloring can be extended in a way that satisfies these constraints. □

As we already mentioned in the introduction, the constraint satisfaction problem for equality constraint languages is always contained in NP. By combining the results obtained in this section and using Theorem 2 and Lemma 1 we therefore proved the following main result of this section.

Theorem 5. *If Γ has no constant unary and no injective binary polymorphism, then CSP(Γ) is NP-complete.*

6 Algorithmic Results

The case that Γ contains a constant unary polymorphism gives rise to trivially tractable constraint satisfaction problems: If an instance of such a constraint satisfaction problem has a solution, then there is also a solution that maps all variables to a single point. In this case an instance of CSP(Γ) is satisfiable if and only if it does not contain a constraint R where R denotes the empty relation in Γ. Clearly, this can be tested efficiently. To finish the classification of the complexity of equality constraint languages we are left with the case that Γ has a binary injective polymorphism.

Lemma 7. *Let Γ be closed under a binary injective polymorphism, and let R be a k-ary relation from Γ. Then for every equivalence relation ρ on $\{1, \ldots, k\}$ (note, that ρ need not be in R) either there is no $\sigma \in R$ that is coarser than ρ, or there exists an equivalence relation $\sigma \in R$ such that σ is coarser than ρ and σ is finer than any $\sigma' \in R$ coarser than ρ. Furthermore, σ can be computed in time $O(k^2|R|)$.*

Proof. First we compute the set R' of equivalence relations of R that are coarser than ρ. The set R' can be computed straightforwardly in time $O(k^2|R|)$ by checking each equivalence relation in R. If R' is empty we are done. Otherwise, because R is closed under intersections, we know that $\sigma = \cap_{\sigma' \in R'} \sigma'$ is in R. It is even in R', since if two equivalence relations are both coarser than another, then so is their intersection. We can find σ with the following procedure.

- We start with an arbitrary equivalence relation τ in R'.
- For each $\sigma' \in R'$, if σ' is finer than τ, then set τ to be σ'.

The procedure clearly runs in time $O(k^2|R|)$. □

Theorem 6. *Let Γ be closed under a binary injective polymorphism, and let S be an instance of CSP(Γ) with n variables and q constraints. Let k be the maximal arity of the constraints, and let m be the maximal number of equivalence relations in the representations for the constraints. Then there is an algorithm that decides the satisfiability of S in time $O(qm(qmk^2 + n))$.*

Proof. We start by assigning each variable a unique value. Then we check whether each constraint is satisfied. If we find an unsatisfied l-ary constraint R, let $x_1, \dots, x_l$ be the variables of that constraint. Let ρ be the equivalence relation on the elements $\{1, \dots, l\}$ that contains all pairs $\{i, j\}$ where x_i got the same value as x_j. Using the algorithm from Lemma 7 we either find that there is no $\sigma \in R$ coarser than ρ, in which case we answer that the problem does not have a solution. Otherwise we find the unique finest equivalence relation $\sigma \in R$. In this case we reassign the values to the variables in the following way: If $\sigma(i, j)$, we assume without loss of generality that $i < j$, and change the value of all variables with the value of x_j to the value of x_i. Finally we restart the procedure with the new assignment for the variables. If all the constraints are satisfied we have computed a solution.

To show the correctness of this algorithm we prove by induction that each of the introduced equalities holds in every solution of the problem. In the beginning we introduced no equality (all the values were mutually different). We introduce an equality only if we find an unsatisfied constraint. In that case we have computed the set of equalities (an equivalence relation) that is contained in every other set of equalities acceptable for the constraint. Because the constraint must be satisfied in every solution we introduce only the equalities that hold in every solution.

Because the set of acceptable equivalence relations is made smaller each time the constraints are not yet satisfied, we have to recompute the assignment at most qm times. Finding the unsatisfied constraint can take $O(qmk^2)$ and changing the assignment can take $O(n)$. Putting the terms together yields the claimed bound on the time complexity. □

Note that the asymptotic running time of the algorithm can be substantially improved by using better data structures.

In the standard case that the signature of Γ is finite, the algorithm clearly establishes the tractability of CSP(Γ) for injective binary polymorphisms, since in this case k and m are bounded by constants that only depend on Γ. If Γ has a countable signature, there are various possibilities to define tractability of CSP(Γ). We refer to the discussion in [6]. The definition of tractability chosen there is to require that for every reduct Γ' of Γ with a finite signature the problem CSP(Γ') is tractable. If Γ has an injective binary polymorphism, this requirement is clearly fulfilled, because we can again use the above algorithm with the same argument. If we allow that the instances contain arbitrary relations from the signature, we have to discuss how to represent the constraints in the instance. For equality constraint languages, one natural candidate to represent the constraints in the instance is the representation that we already used in the formulation of the algorithm: a constraint is represented by a list of equivalence relations on its arguments. Now, the detailed complexity analysis given above shows that we even obtain tractability in the stronger sense where instances might contain arbitrary constraints in the above representation.

7 Conclusion and Remarks

We combine the results of Section 5 and Section 6 and obtain the following.

Theorem 7. *An equality constraint language with template Γ is tractable if Γ has a constant unary or an injective binary polymorphism. Otherwise it is NP-complete.*

In other words, unless P=NP, an equality constraint language with template Γ is tractable if and only if every relation in Γ contains all tuples of the form $(a, \ldots, a)$ for all $a \in \Gamma$, or if all relations are $\cap$-closed.

We would like to conclude with a remark on the relationship of the presented results with questions from universal algebra. The lattice of clones that contain all the permutations is a recent research focus in universal algebra [11, 15], and a full classification seems to be out of reach. However, the lattice of *locally closed* clones that contain the set of all permutations S_ω is considerably simpler. The lattice has a smallest element, the clone that is locally generated by S_ω. Above this clone the lattice has exactly two minimal clones that correspond to the maximally tractable equality constraint languages. Is it possible to give a full description of the locally closed clones that contain all the permutations?

Acknowledgements. We thank the anonymous referees.

References

1. M. Bodirsky. Constraint satisfaction with infinite domains. PhD thesis, Humboldt-Universitat zu Berlin, 2004.
2. M. Bodirsky and J. Nešetřil. Constraint satisfaction with countable homogeneous templates. In *Proceedings of CSL'03*, pages 44–57, 2003.
3. V. G. Bodnarčuk, L. A. Kalužnin, V. N. Kotov, and B. A. Romov. Galois theory for post algebras, part I and II. *Cybernetics*, 5:243–539, 1969.
4. A. Bulatov. Tractable conservative constraint satisfaction problems. In *Proceedings of LICS'03*, pages 321–330, 2003.
5. A. Bulatov, A. Krokhin, and P. Jeavons. The complexity of maximal constraint languages. In *Proceedings of STOC'01*, pages 667–674, 2001.
6. A. Bulatov, A. Krokhin, and P. G. Jeavons. Classifying the complexity of constraints using finite algebras. *SIAM Journal on Computing*, 34:720–742, 2005.
7. P. J. Cameron. *Oligomorphic Permutation Groups.* Cambridge University Press, 1990.
8. R. Dechter and P. van Beek. Local and global relational consistency. *TCS*, 173(1):283–308, 1997.

9. Garey and Johnson. *A Guide to NP-completeness.* CSLI Press, Stanford, 1978.
10. D. Geiger. Closed systems of functions and predicates. *Pacific Journal of Mathematics*, 27:95–100, 1968.
11. L. Heindorf. The maximal clones on countable sets that include all permutations. *Algebra univers.*, 48:209–222, 2002.
12. W. Hodges. *A shorter model theory.* Cambridge University Press, 1997.
13. P. Jeavons, D. Cohen, and M. Gyssens. Closure properties of constraints. *Journal of the ACM*, 44(4):527–548, 1997.
14. M. Krasner. Généralisation et analogues de la théorie de Galois. *Congrés de la Victoire de l'Ass. France avancement des sciences*, pages 54–58, 1945.
15. M. Pinsker. The number of unary clones containing the permutations on an infinite set. *Acta Sci. Math. (Szeged)*, 2005. To appear.
16. R. Pöschel and L. A. Kalužnin. *Funktionen- und Relationenalgebren.* Deutscher Verlag der Wissenschaften, 1979.
17. A. Szendrei. *Clones in universal Algebra.* Seminaire de mathematiques superieures. Les Presses de L'Universite de Montreal, 1986.