Maximal Infinite-Valued Constraint Languages

Manuel Bodirsky [a] Hubie Chen [b] Jan Kára [c] Timo von Oertzen [d]

[a] *LIX, École Polytechnique, Greater Paris*

[b] *Departament de Tecnologies de la Informació i les Comunicacions, Universitat Pompeu Fabra, Barcelona*

[c] *Department of Applied Mathematics, Charles University, Prague*

[d] *Max-Planck-Institute for Human Development, Berlin*

Abstract. We systematically investigate the computational complexity of constraint satisfaction problems for constraint languages over an infinite domain. In particular, we study a generalization of the well-established notion of *maximal constraint languages* from finite to infinite domains. If the constraint language can be defined with an ω-categorical structure, then maximal constraint languages are in one-to-one correspondence to *minimal oligomorphic clones*. Based on this correspondence, we derive general tractability and hardness criteria for the corresponding constraint satisfaction problems.

1 Introduction

One of the research goals in constraint satisfaction is to determine the constraint languages whose constraint satisfaction problem (CSP) can be solved by a polynomial time algorithm (we will call such languages *tractable*), and the languages for which the constraint satisfaction problem is NP-hard. In the last decade, a lot of progress was made in this direction for the case where the domain D of the constraint language is finite. An important stimulant of this research progress has been the observation that for finite D the computational complexity of a constraint satisfaction problem is determined by a set of closure operations, which forms an object known as *polymorphism clone* in universal algebra. The line of research that uses this connection to universal algebra is known as the *algebraic approach* to constraint satisfaction; see e.g. [8] for a recent account.

The complexity classification of CSPs for finite domains D is still not complete. However, based on the algebraic approach, the complexity classification

This is a post-print of a paper published in TCS (410), p. 1684-1693, 2009

for *maximal constraint languages* has been completed recently [6, 7]. Roughly speaking, a constraint language is maximal if it is as large as possible without expressing *all* relations on D. Hence, tractability results for maximal constraint languages are the broadest ones that one can hope for. The notion of a maximal constraint language was previously only used for finite domains D.

Compared to constraint satisfaction with finite domains, there are not as many systematic results for infinite-valued CSPs. One of the outstanding exceptions is the complexity classification for the tractable sub-languages of Allen's interval algebra. Allen's interval algebra is a (binary) constraint language that allows to specify relative positions of intervals over the rational numbers, and is in its unrestricted form NP-complete [1]. However, there are many tractable sub-languages, which were determined in a series of papers, most notably in [18, 19].

Many constraint languages in temporal and spatial reasoning, but also constraint languages studied in computational linguistics and computational biology, are *ω-categorical*. The CSPs for the fragments of Allen's interval algebra, for instance, can all be formulated with ω-categorical constraint languages. The concept of ω-categoricity is of central importance in model theory. It turns out that the algebraic approach to constraint satisfaction can be applied not only to constraint languages over a finite domain, but also to ω-categorical constraint languages [3, 5, 14]. From the model-theoretic point of view, ω-categoricity is a severe restriction. However, the class of CSPs that can be formulated with ω-categorical structures is very rich. It contains for instance all CSPs (for a constraint language over an arbitrary infinite domain) in MMSNP [3], a fragment of existential second order logic, which was introduced in the context of constraint satisfaction in [11]. We will later also see how ω-categorical structures can be used to model problems about solving equations over infinite-dimensional vector spaces.

Contributions. In this paper, we introduce and investigate a notion of maximal constraint languages that applies to infinite domains. For finite domains, our definition of maximal constraint languages essentially coincides with the well-established notion of maximal constraint languages in [6, 7]. We will use the fact that maximal ω-categorical constraint languages are in one-to-one correspondence to minimal oligomorphic clones, and prove that maximal constraint languages are either NP-hard, or have a polymorphism that has one out of three types: either a *unary constant operation*, an *oligopotent quasi majority operation*, or an *oligopotent essential binary operation*. If the polymorphism is of the first type, the constraint satisfaction problem is trivial and tractable. If the polymorphism is of the second type, tractability follows from a result in [3]. Therefore, all maximal constraint languages of unknown computational complexity are preserved by an essential binary operation.

Another main contribution is a strong tractability criterion which shows that ω-categorical constraint languages with a certain binary polymorphism can be solved in polynomial time. This class contains many maximal constraint languages. As demonstrated in Section 8, our condition also captures and extends tractability results in qualitative spatial reasoning, and it provides an universal-algebraic perspective on these results.

2 The Constraint Satisfaction Problem

We first recall fundamental concepts and notation used throughout the text; the book of Hodges [15] might serve as an introduction. A *relational signature* τ is a (here always at most countable) set of *relation symbols* R_i, each associated with an *arity* k_i. A *(relational) structure* Γ over the relational 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. If necessary, we write R^Γ to indicate that we are talking about the relation R belonging to the structure Γ. For simplicity, we denote both a relation symbol and its corresponding relation with the same symbol. For a τ-structure Γ and $R \in \tau$ it is convenient to say that $R(u_1, \ldots, u_k)$ *holds in* Γ iff $(u_1, \ldots, u_k) \in R$. We sometimes use the shortened notation $\overline{x}$ for a vector $x_1, \ldots, x_n$ of any length. Sometimes we do not distinguish between the symbol for a relational structure Γ and its domain D_Γ. If we add relations to a given structure Γ, we call the resulting structure Γ' an *expansion* of Γ, and we call Γ a *reduct* of Γ'.

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^Γ in τ and each n-tuple $\overline{a}$, if $\overline{a} \in R^\Gamma$, then $(f(a_1), \ldots, f(a_n)) \in R^{\Gamma'}$. In this case we say that the mapping f *preserves* the relation R. If there is a homomorphism from Γ to Γ' and a homomorphism from Γ' to Γ, we say that Γ and Γ' are *homomorphically equivalent*. Homomorphisms from Γ to Γ are called *endomorphisms*. A homomorphism is called a *strong homomorphism* if it satisfies the stronger condition that for each n-tuple $\overline{a}$, $\overline{a} \in R^\Gamma$ if and only if $(f(a_1), \ldots, f(a_n)) \in R^{\Gamma'}$. An injective strong homomorphism is called an *embedding*. An *isomorphism* is a surjective embedding. Isomorphisms from Γ to Γ are called *automorphisms*. The set of all automorphisms $\mathrm{Aut}(\Gamma)$ of a structure Γ is a group with respect to composition.

The constraint satisfaction problem. Let Γ be a structure with the relational signature τ. The constraint satisfaction problem (CSP) for Γ is the following computational problem:

CSP(Γ)
INSTANCE: A finite structure S of the same relational signature τ as the template Γ.
QUESTION: Is there a homomorphism from S to Γ?

The structure Γ is called the *template* or the *constraint language* of CSP(Γ). The elements of the finite input structure S are also called the *variables* of the CSP. In order to study the computational complexity of this problem, we have to encode the input structure S as a finite string over the alphabet $\{0, 1\}$. However, if we assume that the signature τ is finite, the exact choice of the representation of the relation symbols does not matter (since the relational signature is fixed and in particular does not grow with the size of the input). For infinite signatures τ, we say that CSP(Γ) is tractable if and only if CSP(Γ′) is tractable for all all reducts Γ′ of Γ having a *finite* signature. This definition is also commonly used for constraint satisfaction over finite domains [8].

To study the computational complexity of the CSP, reductions from one constraint language to another can be described conveniently using the notion of *primitive positive definability* from logic. A first-order formula is called *primitive positive (pp)*, if it has the form $\exists x_1 \dots x_k.\psi_1 \wedge \dots \wedge \psi_l$, where ψ_i is atomic (it might be of the form $x = y$, i.e., we always include equality in first-order logic). The atomic formulas might contain free variables and existentially quantified variables from $x_1, \dots, x_k$. As usual, every formula with k free variables defines a k-ary relation on a structure Γ. Primitive positive definability of relations is an important concept in constraint satisfaction, because pp-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. [8].

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 up to logspace reductions.*

The (universal-) algebraic approach to constraint satisfaction relies on the fact that pp-definability can be characterized by preservation under so-called *polymorphisms*; we introduce these concepts in Section 4.

3 Preliminaries from Model Theory

We first recall fundamental concepts from model theory, which are standard, see e.g. [15]. A relational structure over a countably infinite domain is called

ω-categorical if the first-order theory of Γ, i.e., the set of first-order sentences that is true in Γ, has only one countable model up to isomorphism. The following deep theorem discovered independently by Engeler, Ryll-Nardzewski, and Svenonius (see [15]) describes these structures in algebraic terms.

An *orbit* of a k-tuple $(t_1, \ldots, t_k) \in D^k$ under a permutation group G is the set of all tuples of the form $(\pi(t_1), \ldots, \pi(t_k))$, where π is a permutation from G. A permutation group G is called *oligomorphic*, if for each $k \geq 1$, there are finitely many orbits of k-tuples under G.

Theorem 2 (Engeler, Ryll-Nardzewski, Svenonius; see [15]) *A countable relational structure is ω-categorical if and only if the automorphism group of Γ is oligomorphic. A relation R has a first-order definition in an ω-categorical structure Γ if and only if R is preserved by all automorphisms of Γ.*

An ω-categorical structure Γ is called *model-complete* if every embedding from Γ into Γ preserves all first-order formulas. It is called *homogeneous* (in the literature sometimes also *ultra-homogeneous*), if all isomorphisms between finite induced substructures of Γ can be extended to automorphisms of Γ. It is well-known [15] that an ω-categorical structure is homogeneous if and only if it has *quantifier-elimination*, i.e., every first-order formula is equivalent to a quantifier-free formula over Γ. Homogeneous ω-categorical structures are always model-complete [15]. We say that an ω-categorical structure Γ is a *core*, if every endomorphism of Γ is an embedding. We say that Δ is a *core* of Γ if Δ is a core and homomorphically equivalent to Γ.

Theorem 3 (from [14]) *Every ω-categorical structure Γ has a model-complete core Δ, which is unique up to isomorphism, and which is either finite or ω-categorical. Every relation consisting of a single orbit of k-tuples under the automorphism group of a model-complete core Δ has a primitive positive definition in Δ.*

Since a model-complete core Δ of Γ is unique up to isomorphisms, we call Δ *the core* of Γ. Clearly, Γ and Δ have the same constraint satisfaction problem, and we can therefore always assume that templates of constraint satisfaction problems are model-complete cores. One of the reasons why it is convenient to assume that Γ is a model-complete core is the following.

Lemma 4 (from [14]) *Let Γ be a model-complete ω-categorical core, and let Γ' be the expansion of Γ by finitely many unary singleton relations, i.e., relations of the form $C = \{c\}$. Then CSP(Γ) and CSP(Γ') are polynomial-time equivalent.*

4 Preliminaries from Universal Algebra

To explore the expressive power of a constraint language, we make use of universal algebraic techniques. We give a very short but self-contained introduction to clones on infinite domains.

Let D be an infinite set, and let $O^{(k)}$ be the set of functions from D^k to D, for $k \geq 1$. The symbol O denotes $\bigcup_{k=1}^{\infty} O^{(k)}$. The elements of O will be called *operations* in the following. An operation $\pi \in O^{(k)}$ is called a *projection* if for some fixed $i \in \{1, \ldots, k\}$ and for all k-tuples x we have the identity $\pi(x_1, \ldots, x_k) = x_i$. The *composition* of a k-ary operation f and k operations $g_1, \ldots, g_k$ of arity n is the n-ary operation defined by

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

We say that an operation $f \in O^{(k)}$ 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 $f(t) = g(t)$ for every $t \in B^k$. The set of all operations that are interpolated by F is denoted by $I(F)$.

A subset F of O is called a *clone* if it contains all *projections* and is *closed under composition*. The smallest clone that contains F as a subset is called the clone *generated* by F, and denoted by $G(F)$. A clone F is called a *local clone* (or *locally closed*) if $I(F) = F$. The smallest local clone that contains a set of operations F is called the clone *locally generated* by F, and denoted by $L(F)$. If g is an operation in $L(F)$, we also say that F *locally generates* g. The following fact is mentioned in [22].

Proposition 5 *For all $F \subseteq O$ we have that $L(F) = I(G(F))$.*

Note that $L(F) = G(I(F))$ is in general not true. The connection to the expressive power of constraint languages is as follows. The *(direct–, categorical–, or cross–) product* $\Gamma_1 \times \Gamma_2$ of two relational τ-structures Γ_1 and Γ_2 is a τ-structure on the domain $D_{\Gamma_1} \times D_{\Gamma_2}$. For all relations $R \in \tau$ the relation $R\Big((x_1, y_2), \ldots, (x_k, y_k)\Big)$ holds in $\Gamma_1 \times \Gamma_2$ iff $R(x_1, \ldots, x_k)$ holds in Γ_1 and $R(y_1, \ldots, y_k)$ holds in Γ_2. Homomorphisms from $\Gamma^k = \Gamma \times \ldots \times \Gamma$ to Γ are called *polymorphisms* of Γ. The set of all polymorphisms of a relational structure Γ with domain D, denoted by $\mathrm{Pol}(\Gamma)$, is a local clone (with domain D).

The algebraic approach is based on the following observation, which shows together with Lemma 1 that the computational complexity of a constraint satisfaction problem with template Γ is determined by the polymorphisms of Γ. If F is a clone, we denote by $\mathrm{Inv}(F)$ the set of relations that are preserved by F.

Theorem 6 (from [5]) *Let Γ be an ω-categorical structure. Then a relation is primitive positive definable in Γ if and only if it is preserved by all polymorphisms of Γ. In other words, Inv(Pol(Γ)) is the set of all primitive positive definable relations of Γ.*

This motivates the study of polymorphism clones of ω-categorical structures. A clone F on a countable set D is called *oligomorphic*, if the permutations of D that are contained in F form an oligomorphic permutation group. Theorem 2 asserts that the polymorphism clones of ω-categorical structures are oligomorphic. Conversely, a locally closed oligomorphic clone is the polymorphism clone of an ω-categorical structure; see [2].

An operation of an oligomorphic clone F is called *elementary* if it is locally generated by the permutations in F. Clearly, for finite clones, the elementary operations are the operations that are composed of a projection with a permutation. Note that all endomorphisms of a model-complete ω-categorical core are elementary.

Proposition 7 (from [2]) *If all polymorphisms of an ω-categorical structure Γ are locally generated by the automorphisms of Γ, then every first-order formula is equivalent to a primitive positive formula in Γ.*

We now define several other important properties of k-ary operations. A k-ary operation f is

- *idempotent* iff $f(x,\dots,x) = x$;
- *oligopotent* iff $g(x) := f(x,\dots,x)$ is elementary
- *essentially unary* iff there is a unary operation f_0 such that $f(x_1,\dots,x_k) = f_0(x_i)$ for some $i \in \{1,\dots,k\}$;
- *essential* iff f is not essentially unary;
- a *quasi near-unanimity operation* (short, *qnu-operation*) iff $f(x,\dots,x) = f(x,\dots,x,y) = \cdots = f(x,\dots,x,y,x,\dots,x) = \cdots = f(y,x,\dots,x)$;
- a *quasi majority operation* iff f is a ternary quasi near-unanimity operation;
- a *quasi semiprojection* iff $k \geq 3$ and there is an essentially unary operation g such that $f(x_1,\dots,x_k) = g(x_1,\dots,x_k)$ whenever $|\{x_1,\dots,x_k\}| < k$.

An idempotent quasi majority, quasi near-unanimity, and quasi semiprojection is known as *majority*, *near-unanimity operation*, and *semiprojection*, respectively. If all operations of a clone are elementary, essentially unary, or oligopotent, then we say that the clone is *elementary*, *essentially unary*, or *oligopotent*, respectively. Note that elementary clones are essentially unary.

Minimal Clones. Important questions in universal algebra and useful tools for constraint satisfaction arise from the notion of *minimal clones*. A (proper) subclone F' of a clone F is a clone on the same domain as F such that the

operations of F' form a (proper) subset of the operations of F. An oligomorphic clone F is called *minimal*, if it is non-elementary and every proper oligomorphic subclone is elementary. The following is the oligomorphic analog of a result proved by Rosenberg [21] for clones with a finite domain [1].

Theorem 8 (of [2]) *Every minimal oligomorphic clone F is locally generated by the permutations from F and a non-elementary operation that is of one of the following types*

(1) a unary operation f such that $f(f)$ and the permutations in F locally generate f;
(2) a binary oligopotent operation;
(3) a ternary oligopotent quasi majority operation;
(4) a k-ary oligopotent quasi semi-projection, for $k > 2$.

We also need the following.

Theorem 9 (of [2]) *Let Γ be an ω-categorical model-complete core. If the polymorphism clone F of Γ contains a non-elementary operation, then F also contains a minimal oligomorphic clone.*

5 Hardness Criteria for CSPs

We show that if a constraint language is not preserved by polymorphisms of a special kind, the corresponding constraint satisfaction problem must be NP-hard.

Theorem 10 *Let Γ be an ω-categorical structure. Then either Γ has a finite core, CSP(Γ) is NP-hard, or Γ has a polymorphism f of one of the following types.*

- *an essential binary operation*
- *a ternary quasi majority operation*
- *a k-ary quasi semi-projection, for $k \geq 3$*

PROOF. Let Δ be the (up to isomorphism unique) model-complete core of Γ, and let h be a homomorphism from Γ to Δ (both Δ and h exist due to

[1] Note that one of the five cases presented by Rosenberg can not occur for minimal oligomorphic clones, essentially because ω-categorical model-complete cores can not have a ternary polymorphism that satisfies the identities $f(y,x,x) = f(x,x,y) = f(y,y,y)$; see [2].

Theorem 3). If Δ is a finite τ-structure, there is nothing to show. Otherwise, Δ is ω-categorical (again Theorem 3).

Let F be the polymorphism clone of Δ. If F is elementary, i.e., if F is locally generated by the automorphisms of Δ, then all first-order formulas are equivalent to a pp-formula in Δ (Proposition 7). In particular, this holds for the ternary relation defined by $(x = y \wedge y \neq z) \vee (x \neq y \wedge y = z)$, which has an NP-complete CSP [4] (also see Subsection 8.1). Lemma 1 implies that CSP(Δ) and thus also CSP(Γ) is NP-hard.

Otherwise, F is not elementary, and Theorem 9 implies that F contains a minimal subclone F'. Now, by Theorem 8, the clone F' is locally generated by an operation f that has one out of four types. We can exclude the first case, because Δ is a model-complete core, and therefore all endomorphisms are elementary. For the remaining three cases of Theorem 8, suppose that f is k-ary, and consider the operation g defined by $g(x_1, \ldots, x_k) = f(h(x_1), \ldots, h(x_k))$. It is straightforward to verify that it has again one of the three types, and that it is a polymorphism of Γ. This completes the proof. $\Box$

It was shown in [16] that for constraint languages over finite domains the semiprojections alone do not guarantee tractability. The same holds even for ω-categorical constraint languages and *quasi semiprojections*.

Proposition 11 *Let Γ be an ω-categorical core. If all polymorphisms of Γ are locally generated by quasi semiprojections and the automorphisms of Γ, then CSP(Γ) is NP-hard.*

PROOF. We show this by reducing the problem positive 1-IN-3-3SAT [13] to CSP(Γ). Consider the expansion Γ' of Γ by the two unary singleton relations $A := \{a\}$ and $B := \{b\}$, where a and b are two distinct elements of Γ. Lemma 4 shows that Γ and Γ' have polynomial time equivalent constraint satisfaction problems, so it suffices to show that CSP(Γ') is NP-hard. Since Γ' has strictly fewer polymorphisms than Γ, we have that all the polymorphisms of Γ' are locally generated by quasi semiprojections.

First note that the unary relation $C := \{a, b\}$ is preserved by all quasi semiprojections. Since all quasi semiprojections are at least ternary, every application $f(c_1, \ldots, c_k)$ of a quasi semiprojection f with arguments $c_1, \ldots, c_k$ from $\{a, b\}$ must have two arguments with the same value, and hence f acts like an essentially unary operation on $\{a, b\}$. That is, there is an $i \in \{1, \ldots, k\}$ and a unary operation f_0 such that $f(c_1, \ldots, c_k) = f_0(c_i)$ for all $c_1, \ldots, c_k \in \{a, b\}$. Since the unary operation f_0 must preserve A and B, it must map a to a and b to b, and hence f_0 and f preserve the set C. By Theorem 6 the unary relation C is primitive positive definable in Γ'.

Also consider the ternary relation $N = \{(a,a,b),(a,b,a),(b,a,a)\}$. Similarly to the argument before, we can show that N has a primitive positive definition in Γ' (essentially because all semiprojections behave on the set $\{a,b\}$ like essentially unary operations). By Lemma 1, it suffices to show hardness of CSP(Γ'') where Γ'' is the expansion of Γ' by the relation C and the relation N.

We show hardness of CSP(Γ'') by reduction from 1-IN-3-3SAT. Let S be an instance of positive 1-IN-3-3SAT. We construct an instance of CSP(Γ'') as follows. For each variable in S we introduce a variable x and impose the constraint C on x, i.e., we add x to the unary relation C in the instance we construct. Then every clause in S can be translated to a ternary constraint involving the relation N in the straightforward way, and it is easy to verify that this indeed is a reduction of 1-IN-3-3SAT. □

6 Maximal Constraint Languages

A constraint language Γ is called *complete*, if every first-order definable relation in Γ also has a primitive positive definition in Γ. We call an incomplete constraint language *maximal*, if adding any relation to the language that was not primitive positive definable before turns the constraint language into a complete constraint language. We will see that for ω-categorical structures, maximal constraint languages precisely correspond to minimal locally closed clones.

This definition of maximality coincides with the well-established notion of maximality of constraint languages for finite templates (see [7,8]), if we assume that the constraint language additionally contains for every element $a \in D$ a symbol R_a that denotes the unary relation $\{a\}$. Every CSP with a finite domain is polynomial-time equivalent to the CSP where the language has been expanded by all singleton relations as described above; see [8].

We briefly recall the definition of maximality given in [7]. For finite domains D a constraint language is called *complete* if *every* relation over D has a primitive positive definition over the constraint language. As before, an incomplete constraint language is called *maximal* if adding any relation to the language that was not primitive positive definable before turns the constraint language into a complete constraint language. Once we have singleton relations for all elements in our constraint language, this definition of completeness coincides with the definition shown above, because in such constraint languages every relation has a first-order definition, which is easy to see. Therefore, our definition and the standard definition of maximality essentially coincide on finite templates.

Proposition 12 (of [2]) *Let Γ be an ω-categorical constraint language that is not complete, and let F be the polymorphism clone of Γ. Then the following are equivalent:*

(1) Γ is maximal;
(2) F is minimal, i.e., every proper oligomorphic subclone of F is elementary;
(3) Every non-elementary operation in F, together with the permutations in F, locally generates F.

Theorem 10 now specializes to the following.

Theorem 13 *Let Γ be ω-categorical and maximal. Then one of the following cases applies.*

(1) CSP(Γ) is NP-hard;
(2) Γ has a constant polymorphism, and CSP(Γ) is tractable;
(3) The polymorphism clone of Γ is locally generated by the automorphisms of Γ and an oligopotent essential binary operation;
(4) The polymorphism clone of Γ contains an oligopotent quasi majority operation, and CSP(Γ) is tractable.

PROOF. Suppose that Γ is not a core, i.e., Γ has an endomorphism e that is not an embedding. In particular, e is not elementary. If e is a constant endomorphism, then the following algorithm solves the corresponding trivial CSP. The algorithm first tests whether the instance contains a symbol for an empty relation. If this is the case, the algorithm rejects. Otherwise it can accept the instance, because setting all variables to the constant value of the unary polymorphism e satisfies all constraints.

Proposition 12 implies that the polymorphism clone F of the structure Δ induced by the image of e in Γ is elementary. If the domain of Δ is finite and has more than one element, then NP-completeness of CSP(Δ) follows from results in [8] (because all polymorphisms of Δ are essentially unary). For the case where the domain of Δ is infinite we consider the ternary relation defined by $(x = y \wedge y \neq z) \vee (x \neq y \wedge y = z)$, which has an NP-complete CSP [4] (also see Subsection 8.1). Since F preserves this relation, Theorem 6 implies that this relation has a pp-definition in Δ. By Lemma 1 the problem CSP(Δ) is NP-hard. Thus, CSP(Γ) is also NP-hard.

We now assume that Γ is a core. Theorem 10 shows that if CSP(Γ) is not NP-hard, then Γ has a polymorphism g that is an essential binary operation, a ternary quasi majority, or a k-ary quasi semi-projection. Since Γ is a maximal constraint language, Proposition 12 implies that g together with the automorphisms of Γ locally generates all polymorphisms of Γ. Since Γ is a core, g is oligopotent.

In the case of a quasi semiprojection, Proposition 11 shows that CSP(Γ) is NP-hard. In the case of a quasi majority operation, Theorem 8 in [3] shows that CSP(Γ) can be solved in polynomial time (since a quasi majority is a ternary near-unanimity operation, and languages closed under a quasi near unanimity operation are tractable by Datalog). $\Box$

Note that all the maximal ω-categorical constraint languages of unknown computational complexity are from the third case of Theorem 13.

7 Horn Tractability

In this section we present a general tractability criterion based on preservation under a binary polymorphism, and discuss maximal constraint languages that can be shown to be tractable because they satisfy the criterion.

7.1 Tractability

If $\Gamma = (D; R_1, \dots)$ is a relational structure, we denote by $\hat{\Gamma}$ the expansion of Γ that also contains the complement for each relation in Γ. We say that a relation R has a *quantifier-free Horn definition* in Γ if R can be defined by a quantifier-free first-order formula over the signature of Γ that is in conjunctive normal form in which every clause contains at most one positive literal.

Proposition 14 *Let Γ be an ω-categorical homogeneous structure, and let Δ be a structure with a first-order definition in Γ. If Δ has a polymorphism i which is a strong homomorphism from Γ^2 to Γ, and if CSP($\hat{\Gamma}$) is tractable, then CSP(Δ) is tractable as well.*

PROOF. Let R be a relation from Δ. Because R has a first-order definition in Γ, and because Γ is ω-categorical and homogeneous, it has quantifier-elimination. Therefore, there is a first-order definition ϕ of R in Γ that is quantifier-free. We assume that ϕ is written in conjunctive normal form, and that ϕ is *reduced* in the following sense: whenever we remove a literal or a clause from ϕ, we obtain a formula that is inequivalent to ϕ over Γ. Clearly, we can assume that such a reduced first-order definition ϕ of R exists.

We claim that in this case ϕ is necessarily Horn, i.e., each clause of ϕ contains at most one positive literal. Suppose for contradiction that ϕ contains a clause α that has two positive literals α_1 and α_2. Let ϕ' be the formula ϕ without the literal α_1, and let ϕ'' be the formula ϕ without the literal α_2. Because ϕ is

reduced, there is a tuple a in R that satisfies ϕ but not ϕ', and a tuple b that satisfies ϕ but not ϕ''. Then the tuple $i(a,b)$ does not satisfy α_1, because i is a strong homomorphism from Γ^2 to Γ and b does not satisfy α_1. Similarly, $i(a,b)$ does not satisfy α_2 and also does not satisfy all other literals in α. Hence, i violates ϕ. This contradicts the assumption that i is a polymorphism of Δ.

Finally, we show that if $\mathrm{CSP}(\hat{\Gamma})$ is tractable, then $\mathrm{CSP}(\Delta)$ is tractable as well. We only have to show the tractability of $\mathrm{CSP}(\Delta)$ for structures Δ with a finite signature. As shown in the first part of the proof, each relation in Δ has a reduced first-order definition in Γ, which is Horn. Consider the set of all relations from Γ that are involved in all these definitions; this is again a finite set of relations. Let Γ' be the smallest reduct of Γ that contains all these relations and such that all complements of relations in Γ' are in Γ'. We assume that the relations of the input instances S of $\mathrm{CSP}(\Delta)$ are represented by quantifier-free Horn formulas in reduced form. Note that we can make this assumption because the constraint language Δ is fixed and finite, and also because the constraint language Γ' is fixed and finite.

Now, let S be an instance of $\mathrm{CSP}(\Delta)$ with n variables. The algorithm proceeds as follows.

(1) We consider the instance S' that contains all the clauses from formulas of constraints in S that do not contain negative literals. Note that S' is an instance of $\mathrm{CSP}(\Gamma')$. By assumption, we can decide in polynomial time whether S' has a solution. If S' does not have a solution, the algorithm terminates and reports that S does not have a solution.
(2) We select the negative literals $\neg\beta$ in formulas of constraints from S one by one, and let S'' be the instance of $\mathrm{CSP}(\Gamma')$ obtained from S' by adding $\neg\beta$ (here we use that Γ' also contains the complements of all its relations). We solve S'' with the polynomial-time algorithm for $\mathrm{CSP}(\Gamma')$; if it does not have a solution, we remove $\neg\beta$ from all clauses in the formulas of all the constraints in S. Note that it might be that clauses become empty, and in this case the algorithms reports that S does not have a solution.
(3) We return to the first step of the algorithm, until no literals can be deleted from clauses any more.
(4) The algorithm reports that there is a solution for S.

Clearly, this algorithm works in polynomial time, since there is only a polynomial number (in fact, since the language Γ' and Δ is finite, only a linear number) of literals that can be deleted from formulas defining the constraints in S. It is easy to show by induction that if the algorithms reports that there is no solution for S, then indeed there is no homomorphism from S to Δ. The reason is that whenever a literal $\neg\beta$ is removed from all clauses, then in fact β is implied by the other clauses in S, and therefore removing $\neg\beta$ from all clauses does not affect the solution space of S.

The interesting part is that the final answer in Step 4 of the algorithm is correct. Let n be the number of variables in S, and let Φ be the set of all clauses from formulas of constraints in S at the very final stage of the algorithm. Also consider the negative literals $\neg\gamma_1, \ldots, \neg\gamma_m$ that are finally in the clauses for the constraints in S. For each $\neg\gamma_i$, let $t_k \in D^n$ be an n-tuple that satisfies all clauses without negative literals and where $\neg\gamma_k$ is true, i.e., where γ_k is false. These tuples must exist, since otherwise γ_k would have been true in all solutions, and our algorithm would have jumped from Step 3 to Step 1. The mapping $j : (x_1, x_2, \ldots, x_{m-1}, x_m) \mapsto i(x_1, i(x_2, \ldots, i(x_{m-1}, x_m)\ldots))$ is a strong homomorphism from Γ^m to Γ. It is straightforward to verify that $(j(t_1[1], \ldots, t_m[1]), \ldots, j(t_1[n], \ldots, t_m[n]))$ is a solution for S: negative literals $\neg\gamma_k$ are satisfied because t_k does not satisfy γ_k, and hence γ_k is not satisfied in Γ^m as well. Positive literals from Φ are clearly preserved by j because they hold in all t_k. □

To illustrate this result, we present a simple application for a constraint language over the Boolean domain. Many examples for infinite-valued constraint languages will be presented in Section 8.

Let Γ be the constraint language over the domain $\{0, 1\}$ that contains only the unary relation $\{1\}$. Obviously, $\mathrm{CSP}(\hat{\Gamma})$ is tractable, and the binary mininum operation *min* is a strong homomorphism from Γ^2 to Γ. The proof of Proposition 14 shows that all relations that are preserved by *min* have a Horn definition in Γ (which are precisely the Boolean relations that have a propositional Horn definition in the usual sense). Thus, Proposition 14 implies the well-known fact that the Horn satisfiability problem can be solved in polynomial time.

7.2 Maximality

If Γ^2 is *isomorphic* to Γ, then the isomorphism i is in particular a strong homomorphism from Γ^2 to Γ, and we have already one condition for the tractability result of Proposition 14. In this case, and if Γ is model-complete, we can show that the set of all relations preserved by i forms a maximal constraint language (and hence, expanding Γ by any first-order definable relation that does not have a primitive positive definition in Γ results in a constraint language with an NP-hard constraint satisfaction problem). We first observe the following universal algebraic properties of the binary operation i and the endomorphisms of Γ.

Lemma 15 *Let i be an isomorphism between Γ^2 and Γ. Then the following statements hold.*

T1 *for all* $\alpha, \beta \in Aut(\Gamma)$ *we have* $\gamma \in Aut(\Gamma)$ *such that* $\gamma(x) = \alpha(\beta(x))$
T2 *for all* $\alpha \in Aut(\Gamma)$ *we have* $\beta \in Aut(\Gamma)$ *such that* $i(\alpha(x), y) = \beta(i(x, y))$
T2' *for all* $\alpha \in Aut(\Gamma)$ *we have* $\beta \in Aut(\Gamma)$ *such that* $i(x, \alpha(y)) = \beta(i(x, y))$
T3 *there is* $\alpha \in Aut(\Gamma)$ *such that* $i(i(x, y), z) = \alpha(i(x, i(y, z)))$
T3' *there is* $\alpha \in Aut(\Gamma)$ *such that* $i(x, i(y, z)) = \alpha(i(i(x, y), z))$
T4 *there is* $\alpha \in Aut(\Gamma)$ *such that* $i(x, y) = \alpha(i(y, x))$

PROOF. The proof of the first property is trivial (the set of automorphisms of any structure is a group). For the second property, observe that the operation j defined by $i(\alpha(x), y)$ is an isomorphism from Γ^2 to Γ as well; let j^{-1} be its inverse. Then $i(j^{-1})$ is an isomorphism between Γ and Γ, and its inverse $\beta = j(i^{-1})$ is an automorphism β of Γ such that $i(\alpha(x), y) = \beta(i(x, y))$. The other properties can be verified similarly. □

To show that i and the automorphisms of Γ locally generate a minimal oligomorphic clone, we need the following lemma.

Lemma 16 *Let* Γ *be a model-complete* ω*-categorical structure, and let* i *be an isomorphism between* Γ^2 *and* Γ*. Then for every* k*-ary operation* g *generated by* $\{i\} \cup Aut(\Gamma)$ *and every finite subset* A *of the domain of* Γ *either*

(1) there exists an $l \in \{1, \ldots, k\}$ *and an* $\alpha \in Aut(\Gamma)$ *such that* $g(x_1, \ldots, x_k) = \alpha(x_l)$ *for all* $x_1, \ldots, x_k \in A$*, or*
(2) there are distinct $l_1, l_2 \in \{1, \ldots, k\}$*,* $s_1, \ldots, s_k \in \{l_1, l_2\}$*, and an* $\alpha \in Aut(\Gamma)$ *such that* $g(x_{s_1}, \ldots, x_{s_k}) = \alpha(i(x_{l_1}, x_{l_2}))$ *for all* $x_{l_1}, x_{l_2} \in A$*.*

PROOF. Let F be $\{i\} \cup \mathrm{Aut}(\Gamma)$. Since $g \in G(F)$, there is a composition $T(x_1, \ldots, x_k)$ of the operations in F that defines $g(x_1, \ldots, x_k)$. We can apply T2 and T2' until $T(x_1, \ldots, x_k)$ has been transformed to an expression of the form $\alpha_1(\alpha_2(\ldots \alpha_s(T'(x_1, \ldots, x_k))\ldots))$ where $T'(x_1, \ldots, x_k)$ does not contain unary function symbols. Then we repeatedly apply T1 and rewrite the expression to $\beta(T'(x_1, \ldots, x_k))$ for some $\beta \in Aut(\Gamma)$. In the same way, we move unary function symbols to the front during the following transformations, and β will always denote the resulting unary function symbol at the front.

Let A be any finite subset of the domain of Γ. If there is only one variable x_l that appears in T', then g is essentially unary, and it is clear that g is an embedding of Γ into Γ. Because Γ is model-complete, g must be elementary. Therefore, there exists an automorphism α of Γ such that $g(x_1, \ldots, x_k) = \alpha(x_l)$ for all $x_1, \ldots, x_k \in A$.

Otherwise, assume without loss of generality that both variables x_1 and x_2 do appear in T' (otherwise we permute arguments of g), and consider the equa-

tion $g(x, y, \ldots, y) = \beta(T'(x, y, \ldots, y))$, which is implied by the previous equation. We can apply T3 (and T1,T2,T2') until we obtain an expression of the form $\beta(i(u_1, i(u_2, \ldots i(u_{s-2}, i(u_{s-1}, u_s)) \ldots)))$ where $u_1, u_2, \ldots, u_{s-2}, u_{s-1}, u_s \in \{x, y\}$.

We now prove the statement of the lemma by induction on s. We first make a case distinction to deal with the inductive step for $s > 2$. Note that because the unary operation defined by $i(x, x)$ is elementary, there exists an automorphism α of Γ such that

$$\alpha(x) = i(x, x) \text{ for all } x \in A \,. \tag{1}$$

If $u_{s-1} = u_s = x$ or $u_{s-1} = u_s = y$, we replace $i(u_{s-1}, u_s)$ by $\alpha(u_{s-1})$, and after moving the automorphisms to the front we obtain an expression of the form $\beta(i(u_1, i(u_2, \ldots i(u_{s-2}, u_{s-1}) \ldots)))$, which is equal to $g(x, y, \ldots, y)$ for all $x, y \in A$. If $u_{s-1} \neq u_s$, then either $u_{s-2} = u_{s-1}$ or $u_{s-2} = u_s$. In the first case, we apply T3' and Equation 1 and after moving automorphisms to the front obtain a term of the form $\beta(i(u_1, i(u_2, \ldots i(u_{s-2}, u_s) \ldots)))$. In the second case, we first apply T4, then T3', and finally Equation 1, and after moving automorphisms to the front obtain a term of the form $\beta(i(u_1, i(u_2, \ldots i(u_{s-2}, u_{s-1}) \ldots)))$. For the base case where $s = 2$, we have either $g(x, y, \ldots, y) = \alpha(i(x, y))$ or $g(x, y, \ldots, y) = \alpha(i(y, x))$. In the second case, we apply T4 a last time, and have shown that (2) holds in both cases. □

Proposition 17 *Let Γ be an ω-categorical model-complete structure.* [2] *If there is an isomorphism i between Γ^2 and Γ, then the constraint language of all relations preserved by i and by the automorphisms of Γ is maximal.*

PROOF. Let F' be $\{i\} \cup \mathrm{Aut}(\Gamma)$, and let F be the clone that is locally generated by F', i.e., $F = L(F')$. Since i is essential, Γ is not complete. By Proposition 12, we have to show that every operation in F is either locally generated by the automorphisms of Γ or, with the automorphisms of Γ, locally generates F.

Let f be a k-ary operation in F. By Proposition 5, $f \in I(G(F'))$. Hence, for every finite subset of elements A there is a k-ary operation $g \in G(F')$ such that $f(x) = g(x)$ for all $x \in A^k$, and g satisfies either (1) or (2) in Lemma 16.

Note that if A has at least two elements, then either all operations $g \in G(F')$ with $f(x) = g(x)$ for all $x \in A^k$ satisfy Property (1), or all such operations satisfy (2) (because if g satisfies Property (1) then $|g(A^k)| = |A|$, whereas if g

[2] In the published version of the paper, the statement of this proposition by mistake assumes that Γ is also a core; as we show here, this assumption is not necessary.

satisfies Property (2) then $|g(A^k)| > |A|$). In the first case, we will say that f *satisfies (1) on* A, and in the second that f *satisfies (2) on* A.

It can not be that there are finite subsets A, B of cardinality larger than one such that f satisfies (1) on A and satisfies (2) on B. Otherwise, consider the finite set $A \cup B$. If f satisfies (1) on $A \cup B$, then it satisfies (1) also on B, a contradiction, and if it satisfies (2) on $A \cup B$, then it also satisfies (2) on A, also a contradiction.

If f satisfies (1) for all finite subsets A, then f is locally generated by the automorphisms of Γ. If f satisfies (2) for all finite subsets A, we claim that f and the automorphisms of Γ locally generate i. Let B be a finite subset of the domain of Γ. As we have seen, there exist distinct $l_1, l_2 \in \{1, \ldots, n\}$, $s_1, \ldots, s_k \in \{l_1, l_2\}$, and $\alpha \in Aut(\Gamma)$ such that $f(x_{s_1}, \ldots, x_{s_k}) = \alpha(i(x_{l_1}, x_{l_2}))$ for all $x_{l_1}, x_{l_2} \in B$. But then $\alpha^{-1}(f(x_{s_1}, \ldots, x_{s_k})) = i(x_{l_1}, x_{l_2})$ for all $x_{l_1}, x_{l_2} \in B$, and therefore f locally generates i. Proposition 12 shows that $F = L(F') = I(G(F'))$ is a minimal oligomophic clone. □

Combining Proposition 14 and Proposition 17, we obtain the following result.

Theorem 18 *Let* Γ *be an* ω*-categorical homogeneous structure. If there is an isomorphism* i *between* Γ^2 *and* Γ*, then*

- *the binary operation* $i : D^2 \to D$ *and the automorphisms of* Γ *locally generate a minimal oligomorphic clone* F*; hence,* $\Delta = Inv(F)$ *is a maximal constraint language;*
- *all relations in* Δ *have a quantifier-free Horn definition in* Γ*;*
- *if CSP*$(\hat{\Gamma})$ *is tractable, then CSP*(Δ) *is tractable as well.*

Aplications of Theorem 18 for concrete constraint languages and computational problems can be found in Section 8. We would like to remark that our tractability result is a new application of the universal-algebraic approach to constraint satisfaction, and indeed we have linked tractability of constraint languages to the existence of a binary polymorphism that locally generates a minimal oligomorphic clone. Note that the maximality result as stated in Theorem 18 is specific to infinite domains, because for constraint languages Γ over a finite domain Γ^2 cannot possibly be isomorphic to Γ. For infinite structures Γ the situation that Γ is isomorphic to Γ^2 is not so rare; it is e.g. well-known that the set of models of a universal first-order Horn theory is preserved under direct products [15].

8 Applications

8.1 Equality Constraint Languages

Our first example is a restricted subclass of constraint languages, called *equality constraint languages* [4], which are all constraint languages that are preserved by all permutations of the domain. In this class, there are two maximal languages, which are both tractable, and are hence the largest tractable languages[3]. The tractability of one of these maximal languages follows from Theorem 18.

As a base set, we can take any infinite set, and we use $\mathbb{N}$ for simplicity. We only consider relations that are preserved by *all* permutations of $\mathbb{N}$. Such relations will always be definable by a boolean combination of atomic formulas of the form $x = y$, and a constraint language that consists of such relations only we call an *equality constraint language*; see [4]. One example of an NP-hard constraint language in this class consists of the ternary relation R defined by $(x = y \wedge y \neq z) \vee (x \neq y \wedge y = z)$. Examples of a tractable constraint language from this class consists of the 6-ary relation P defined by $(x = y \wedge u = v) \rightarrow a = b$ and the 4-ary relation Q defined by $(x \neq y) \vee (u \neq v)$.

It was shown in [4] that there are exactly two largest tractable equality constraint languages. One is the language L_c that consists of all relations that are closed under a constant unary operation, and the other language L_i consists of all relations that are closed under a (any!) binary injective operation i. The relations P and Q shown above are examples of relations preserved by all binary injective operations.

We now demonstrate how to apply Theorem 18 to show that L_i is a maximal constraint language, and that $\mathrm{CSP}(L_i)$ is tractable. Let Γ be the structure $(\mathbb{N}, =)$; this structure is clearly ω-categorical and homogeneous. Then $\hat{\Gamma}$ is the structure $(\mathbb{N}, =, \neq)$. Clearly, $\mathrm{CSP}(\hat{\Gamma})$ is tractable. It is also clear that Γ^2 is isomorphic to Γ; in fact, any binary injective operation together with the permutations of $\mathbb{N}$ locally generates isomorphisms between Γ^2 and Γ. We can therefore apply Theorem 18 and obtain that L_i is maximal and that $\mathrm{CSP}(L_i)$ is tractable.

[3] Note the difference between tractable maximal constraint languages, and largest tractable constraint languages. The latter are constraint languages where adding any constraint that was not primitive positive before makes the language intractable. The former are simply constraint languages that are maximal *and* tractable.

Solving equations with equalities and disequalities for an infinite-dimensional vector space over a finite field can be formulated as a constraint satisfaction problem with an ω-categorical template.

Let F_q be a finite field with elements $s_0, \ldots, s_{q-1}$ and let V be a countably infinite vector space over F_q. Then V is unique up to isomorphism [15]. We consider the following relational structure $\Gamma_V := (V, R^+)$ where R^+ is the ternary relation defined by $R^+(x, y, z) \equiv (x + y = z)$. The structure Γ_V is homogeneous and ω-categorical [15]. Hence, its automorphism group is oligomorphic; in fact, the automorphism group is one of the groups known as the *classical infinite groups*, which have many remarkable properties [12]. Clearly, the constraint satisfaction problem for Γ_V is trivial, because it has a constant endomorphism. But it is not difficult to show that the expansion $\hat{\Gamma}_V = (V, R^+, \neg R^+)$ of Γ_V (where $\neg R^+$ denotes the complement of R^+, i.e., the ternary disequality relation defined by $x + y \neq z$) is a core.

The idea how to solve an instance S of $\mathrm{CSP}(\hat{\Gamma}_V)$ in polynomial time is to eliminate for each constraint $R^+(x, y, z)$ in S the variable z from the remaining constraints in S, similarly as in the Gaussian elimination algorithm. That is, for every constraint where z occurs we replace z by $x + y$, and simplify the resulting equality or disequality in the usual way. If after the elimination of a variable x a disequality constraint in S reduces to $0 \neq 0$, the algorithm reports that S is unsatisfiable. Otherwise, we eventually end up with a set of disequality constraints, and no more equality constraints. In this case, the algorithm reports that S has a solution. This is correct, because if $y_1, \ldots, y_k$ are the remaining variables, then the mapping that sends y_i to a vector in V that has a zero entry at all coordinates except for the i-th coordinate where it has an entry distinct from zero is clearly a solution to S.

It is well-known that V^2, the direct product of the algebraic object V with itself, is isomorphic to V: the function i_V that maps the two vectors $(a_1, a_2, \ldots) \in V$ and $(b_1, b_2, \ldots) \in V$ to $(a_1, b_1, a_2, b_2, \ldots) \in V$ is such an isomorphism between V^2 and V, and i_V is also an isomorphism between Γ_V^2 and Γ_V. Let C_V be the oligomorphic clone that is locally generated by i_V and the automorphisms of Γ_V. Then Theorem 18 implies that $\mathrm{Inv}(C_V)$ is a maximal constraint language. Since $\mathrm{CSP}(\hat{\Gamma}_V)$ is tractable as well, we can apply Theorem 18 and find that $\mathrm{CSP}(\mathrm{Inv}(C_V))$ is tractable. This is an interesting result, which says that we can efficiently decide whether a given set of Horn clauses of equations for infinite-dimensional vector spaces over a finite field has a solution.

8.3 Spatial Reasoning

One of the most fundamental spatial reasoning formalisms is the RCC-5 calculus (also known as the containment algebra in the theory of relation algebras [9]). The largest tractable sublanguages of the binary constraint language for RCC-5 have been determined in [17, 20]. Let $\mathbb{B}_0$ be the countable atomless boolean ring without an identity element. This structure is unique up to isomorphism and ω-categorical, see for example [10]. It is straightforward to verify that $(\mathbb{B}_0)^2$ is isomorphic to $\mathbb{B}_0$ (because $(\mathbb{B}_0)^2$ is again a countable atomless boolean ring without an identity element). We are interpreting the elements of this boolean ring as non-empty sets (regions), where $A + B$ denotes the symmetric difference, and $A \cdot B$ the intersection of A and B. Now, consider the relational structure Σ over the domain of $\mathbb{B}_0$ with the two binary relations called DR and PP (these are the traditional names in RCC-5). With the interpretation of the elements of $\mathbb{B}_0$ being sets, they are defined as follows: $\mathtt{DR}(X, Y)$ iff $X \cap Y = \emptyset$ and $\mathtt{PP}(X, Y)$ iff $X \subset Y$.

The constraint satisfaction problem for $\mathrm{CSP}(\hat{\Sigma})$ is in P [20]. Σ^2 is isomorphic to Σ as well, and it again follows by Theorem 18 that the constraint language whose relations have a Horn definition in Σ is tractable. We would like to remark that Nebel and Renz [20] determined a largest tractable fragment of RCC-5, and that it follows from the result in [17] that this fragment is the unique largest tractable fragment that contains the basic relations DR and PP. But note that RCC-5 only contains binary relations, whereas our maximal language contains relations of arbitrary arity.

8.4 Temporal Reasoning

We present another maximal constraint language from the field of temporal reasoning. It will serve us as an example of a maximal constraint language whose polymorphism clone is locally generated by a binary injective operation, but which has an NP-complete CSP.

One of the basic structures for temporal reasoning is $(\mathbb{Q}, <)$, the set of rational numbers, ordered by $<$. This structure is an unbounded and dense linear order on countably many vertices, and it is uniquely described by these properties, up to isomorphism. Note that $(\mathbb{Q}, <)^2$ is clearly *not* isomorphic to $(\mathbb{Q}, <)$.

Consider a binary operation lex on $\mathbb{Q}$ satisfying $\mathrm{lex}(a, b) < \mathrm{lex}(a', b')$ if either $a < a'$, or $a = a'$ and $b < b'$. Note that every operation lex satisfying these conditions is by definition injective. Let F be the clone generated by lex and the automorphisms of $(\mathbb{Q}, <)$. The problem CSP(Inv(F)) is NP-complete. This is because all operations in F preserve the Betweenness relation defined by the

formula $(x < y < z) \vee (z < y < x)$, and the CSP for the Betweenness relation is a well-known NP-complete problem [13]. It is not hard to show that F is a minimal oligomorphic clone.

References

[1] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[2] M. Bodirsky and H. Chen. Oligomorphic clones. *Algebra Universalis*, 57(1):109–125, 2007.

[3] M. Bodirsky and V. Dalmau. Datalog and constraint satisfaction with infinite templates. In *Proceedings of STACS'06*, pages 646–659, 2006.

[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] A. Bulatov. A graph of a relational structure and constraint satisfaction problems. In *Proceedings of LICS'04, Turku, Finland*, 2004.

[7] A. Bulatov, A. Krokhin, and P. Jeavons. The complexity of maximal constraint languages. In *Proceedings of STOC'01*, pages 667–674, 2001.

[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] I. Duentsch. Relation algebras and their application in temporal and spatial reasoning. *Artificial Intelligence Review*, 23:315–357, 2005.

[10] D. Evans. Examples of $\aleph_0$-categorical structures. *In 'Automorphisms of first-order structures', R. Kaye and H.D. Macpherson, Oxford University Press*, pages 33–72, 1994.

[11] T. Feder and M. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory. *SIAM Journal on Computing*, 28:57–104, 1999.

[12] T. Gardener. Infinite dimensional classical groups. *J. London Math. Soc*, 51:219–229, 1995.

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

[14] P. Hell and J. Nesetril. The core of a graph. *Discrete Math.*, 109:117–126, 1992.

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

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

[17] P. Jonsson and T. Drakengren. A complete classification of tractability in RCC-5. *J. Artif. Intell. Res.*, 6:211–221, 1997.

[18] A. Krokhin, P. Jeavons, and P. Jonsson. Reasoning about temporal relations: The tractable subalgebras of Allen's interval algebra. *JACM*, 50(5):591–640, 2003.

[19] B. Nebel and H.-J. Bürckert. Reasoning about temporal relations: A maximal tractable subclass of Allen's interval algebra. *JACM*, 42(1):43–66, 1995.

[20] J. Renz and B. Nebel. On the complexity of qualitative spatial reasoning: A maximal tractable fragment of the region connection calculus. *Artif. Intell.*, 108(1-2):69–123, 1999.

[21] I. G. Rosenberg. Minimal clones I: the five types. *Lectures in Universal Algebra (Proc. Conf. Szeged, 1983), Colloq. Math. Soc. J. Bolyai*, 43:405–427, 1986.

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