The Complexity of Temporal Constraint Satisfaction Problems

Manuel Bodirsky
Laboratoire d'informatique (LIX)
École polytechnique
France
bodirsky@lix.polytechnique.fr

Jan Kára*
Department of Applied Mathematics
Faculty of Mathematics and Physics
Charles University
Czech Republic
kara@kam.mff.cuni.cz

ABSTRACT

A *temporal constraint language* is a set of relations that has a first-order definition in $(\mathbb{Q}, <)$, the dense linear order of the rational numbers. We present a complete complexity classification of the constraint satisfaction problem (CSP) for temporal constraint languages: if the constraint language is contained in one out of nine temporal constraint languages, then the CSP can be solved in polynomial time; otherwise, the CSP is NP-complete. Our proof combines model-theoretic concepts with techniques from universal algebra, and also applies the so-called product Ramsey theorem, which we believe will be useful in similar contexts of constraint satisfaction complexity classification.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Computations on discrete structures*

General Terms

Theory, Algorithms

Keywords

Constraint satisfaction, Temporal reasoning, Complexity

1. INTRODUCTION

A *temporal constraint language* Γ is a relational structure $(\mathbb{Q}; R_1, R_2, \dots)$ with a first-order definition in $(\mathbb{Q}, <)$, the rational numbers with the dense linear order. A first-order sentence is called *primitive positive* if it is of the form

$$\exists x_1, \dots, x_n . \psi_1 \wedge \cdots \wedge \psi_p \ ,$$

where $\psi_1, \dots, \psi_p$ are atomic formulas of the form $R(x_{i_1}, \dots, x_{i_k})$. The *constraint satisfaction problem (CSP)* for a temporal constraint language Γ is the computational problem to decide for a given primitive positive sentence Φ whether Φ is true or false in Γ.

Several famous NP-complete problems can be formulated as temporal CSPs. For example, if the constraint language Γ only contains the ternary relation $Betw = \{(x, y, z) \in \mathbb{Q}^3 \mid (x < y < z) \vee (z < y < x)\}$, then the corresponding CSP becomes the Betweenness problem listed in the book of Garey and Johnson [18]. Similarly, the Cyclic Ordering problem can be formulated as a temporal CSP [18].

One of the most fundamental temporal constraint languages is $(\mathbb{Q}; \leq, <, \neq, =)$, the so-called Point Algebra. The constraint satisfaction problem for the Point Algebra is easily seen to be tractable [23]. A considerably larger temporal constraint language is the set of Ord-Horn relations, introduced by Nebel and Bürkert [24]. The CSP for Ord-Horn can be solved by resolution in polynomial time.

In this paper we present a complete classification of the computational complexity of the constraint satisfaction problem for temporal constraint languages. The CSP can be solved in polynomial time if the temporal constraint language is contained in one out of nine temporal constraint languages; otherwise the problem is NP-complete. Two of these languages properly contain all Ord-Horn relations [4].

A similar classification result was obtained by Schaefer [27] for boolean constraint languages, i.e., relational structures over a two element set where each relation can be defined by a propositional formula. Schaefer showed that the CSP for a boolean constraint language is tractable if the language is contained in one out of six boolean constraint languages; otherwise the CSP is NP-complete.

The question whether such a complexity dichotomy holds for all constraint languages over a *finite* domain [17] is one of the major open research problems in the area. In the last decade, a strong connection of this problem to central and deep questions in universal algebra has stimulated further activity [7–12,16,21,22]; the starting point of this connection is the observation that the complexity of the CSP is fully described by the so-called polymorphisms of the constraint language.

The techniques that we apply to study temporal CSPs take their impetus from this algebraic approach. In order to use polymorphisms for constraint languages over $\mathbb{Q}$, we need fundamental concepts from model theory, as in [1,2,5]. Another important ingredient is Cameron's classification of

*The author is supported by Project 1M0021620808 of the Ministry of Education of the Czech Republic.

temporal constraint languages: up to first-order interdefinability, there are exactly five different temporal constraint languages. First-order definability does not preserve the complexity of the corresponding CSP (this is why we need polymorphisms and universal algebra), but still Cameron's result turns out to be useful in our proof of the complexity dichotomy. In particular, it allows us to perform the essential part of the classification with *binary* polymorphisms only; the techniques how this is done are another major contribution of our work. Finally, in the combinatorial part of the proofs we also apply the so-called product Ramsey theorem; we believe that this theorem can be applied in a similar way for complexity classifications of other classes of constraint languages over infinite domains. The proof techniques, which are part of the contributions of this paper, become visible from the top-level architecture of the classification proof presented in this extended abstract. For proof details we kindly refer the reader to the full version of the paper.

2. MODEL-THEORETIC PRELIMINARIES

A temporal constraint language $\Gamma = (\mathbb{Q}, R_1, R_2, \dots)$ is a structure over a relational signature with a first-order definition in $(\mathbb{Q}, <)$, the dense linear order of the rational numbers. This is, for every relation R_i of Γ of arity k_i there is a first-order formula $\phi(x_1, \dots, x_{k_i})$ with k_i free variables $x_1, \dots, x_{k_i}$ that defines R_i over $(\mathbb{Q}, <)$ in the usual way. In particular, the only atomic formulas of ϕ are of the form $x < y$ (or of the form $x = y$, equality is for us always included in first-order logic; but this does not make a difference here).

The structure $(\mathbb{Q}, <)$ has notable model-theoretical properties, which are important in this paper. The *first-order theory* of a relational structure is the set of all first-order sentences that is true in Γ.

Definition 1. *A relational structure Γ over a countable domain is called* ω-categorical *if all countable models of the first order theory of Γ are isomorphic to Γ.*

The structure $(\mathbb{Q}, <)$ is ω-categorical; this is due to Kantor.

Lemma 1 (see e.g. [20]). *If Γ is ω-categorical, and Δ has a first-order definition in Γ, then Δ is also ω-categorical.*

As a consequence, all temporal constraint languages are ω-categorical. There is also an algebraic characterization of ω-categoricity. An automorphism of a relational structure Γ is an isomorphism between Γ and Γ. The set of all automorphisms of Γ forms a permutation group $\mathrm{Aut}(\Gamma)$ on the domain D of Γ. The *orbit of a k-tuple $t \in D^k$* in Γ is the set $\{\alpha(t) \mid \alpha \in Aut(\Gamma)\}$. The *orbit of a k-set $S \subset D$* (S is a k-set if $|S| = k$) in Γ is the set $\{\alpha(S) \mid \alpha \in Aut(\Gamma)\}$. A permutation group is called *oligomorphic* if for every $k \geq 1$ there is only a finite number of distinct orbits of k-tuples over D. The theorem of Engeler, Svenonius, and Ryll-Nardzewski says that a relational structure Γ is ω-categorical if and only if the automorphism group of Γ is oligomorphic. Automorphism groups of temporal constraint languages even satisfy a stronger property. A permutation group is called *highly set-transitive* if for all $k \geq 1$ there is only one orbit of k-sets.

Another important model-theoretic property of $(\mathbb{Q}, <)$ is that it is *homogeneous*; a relational structure Γ is called *homogeneous* if every isomorphism between induced substructures of Γ can be extended to an automorphism of Γ.

We now quote the classical result of Cameron [13] that describes temporal constraint languages *up to first-order interdefinability*. We say that two structures Γ and Δ are *first-order interdefinable* if Γ has a first-order definition in Δ and Δ has a first-order definition in Γ.

Theorem 2. *Let Γ be a temporal constraint language. Then Γ is first-order interdefinable with exactly one out of the following five homogeneous structures.*

- *The dense linear order $(\mathbb{Q}, <)$ itself,*
- *The structure $(\mathbb{Q}, Betw)$, where Betw is the relation* $\{(x, y, z) \in \mathbb{Q}^3 \mid (x < y < z) \vee (z < y < x)\}$
- *The structure $(\mathbb{Q}, Cycl)$, where Cycl is the relation* $\{(x, y, z) \mid (x < y < z) \vee (y < z < x) \vee (z < x < y)\}$,
- *The structure $(\mathbb{Q}, Sep)$, where Sep is the relation* $\{(x_1, y_1, x_2, y_2) \mid (x_1 < x_2 < y_1 < y_2) \vee (x_1 < y_2 < y_1 < x_2) \vee (y_1 < x_2 < x_1 < y_2) \vee (y_1 < y_2 < x_1 < x_2) \vee (x_2 < x_1 < y_2 < y_1) \vee (x_2 < y_1 < y_2 < x_1) \vee (y_2 < x_1 < x_2 < y_1) \vee (y_2 < y_1 < x_2 < x_1)\}$,
- *The structure $(\mathbb{Q}, \emptyset)$.*

The relation *Sep* is the so-called *separation relation*; note that $Sep(x_1, y_1, x_2, y_2)$ holds for elements $x_1, y_1, x_2, y_2 \in \mathbb{Q}$ iff all four points x_1, y_1, x_2, y_2 are distinct and the smallest interval over $\mathbb{Q}$ containing x_1, y_1 overlaps with the smallest interval containing x_2, y_2.

The following theorem by Cameron [13] was his original motivation for the investigation of the structures with a first-order definition in $(\mathbb{Q}, <)$.

Theorem 3. *A relational structure has a highly set-transitive automorphism group if and only if it is a temporal constraint language.*

3. THE CONSTRAINT SATISFACTION PROBLEM

A first-order formula is called *primitive positive* if it is of the form

$$\exists x_1, \dots, x_n . \psi_1 \wedge \dots \wedge \psi_p \ ,$$

where each formula $\psi_1, \dots, \psi_p$ is atomic, i.e., of the form $R(y_1, \dots, y_k)$ where $y_1, \dots, y_k$ might be either variables from $x_1, \dots, x_n$ or free. The constraint satisfaction problem for a constraint language Γ is the following computational problem, denoted by $\mathrm{CSP}(\Gamma)$. We are given a primitive positive sentence Φ (i.e., a primitive positive formula without free variables) where all relation symbols are relation symbols for the relations in Γ, and the question is whether Φ is true in Γ. If $\Gamma = (\mathbb{Q}, R)$, we also write $\mathrm{CSP}(R)$ instead of $\mathrm{CSP}((\mathbb{Q}, R))$.

Note that for infinite constraint languages the complexity of $\mathrm{CSP}(\Gamma)$ depends also on the representation of relations from Γ. We set aside this representation issue, as infinite languages are not in the main focus of this paper. Let us just mention that for infinite languages the algorithmic results hold for example for the representation of relations by a set of tuples, where we take one tuple from each orbit of tuples in $Aut(\mathbb{Q}, <)$ forming the relation. The hardness results hold for finite and infinite constraint languages regardless of the representation.

The conjuncts $\psi_1, \ldots, \psi_p$ of an instance Φ of the CSP are also called *constraints*. Hence, in this paper a constraint is a syntactic object (an atomic formula). Note that an instance of the CSP is fully described by its set of constraints. The set of variables that appears in a first-order formula Φ is denoted by $V(\Phi)$. A *solution* for an instance Φ of the CSP is a mapping $s : V(\Phi) \to \mathbb{Q}$ that satisfies all the constraints of Φ. The following is easy to see.

Proposition 4. *For all temporal constraint languages Γ the problem CSP(Γ) is in NP.*

The following is an essential tool to establish hardness results for the CSP. A k-ary relation is called *primitive positive definable* if there exists a primitive positive formula with k free variables that defines R. Lemma 5 says that primitive positive definable relations can be 'simulated' in constraint satisfaction problems [12].

Lemma 5. *Let $\Gamma = (\mathbb{Q}, R_1, R_2, \ldots)$ be a constraint language, and let R be a relation that has a primitive positive definition in Γ. Then CSP(Γ) and CSP$((\mathbb{Q}, R, R_1, R_2, \ldots))$ are polynomial-time equivalent.*

4. ALGEBRAIC PRELIMINARIES

Primitive positive definability can be characterized by preservation under so-called *polymorphisms* – this is the starting point of the so-called *(universal-) algebraic approach* to constraint satisfaction (see e.g. [10, 12]).

We say that a k-ary function (also called *operation*) $f : D^k \to D$ *preserves* an m-ary relation $R \subseteq D^m$ if whenever $R(a^i_1, \ldots, a^i_m)$ holds for all $1 \le i \le k$, then $R\big(f(a^1_1, \ldots, a^k_1), \ldots, f(a^1_m, \ldots, a^k_m)\big)$ holds as well. If an operation f does not preserve a relation R, we say that f *violates* R. If f preserves all relations of a relational structure Γ, we say that f is a *polymorphism* of Γ. The unary polymorphisms are called *endomorphisms* of Γ. Note that the automorphisms of Γ are bijective endomorphisms that preserve all relations and their complements.

The set of all polymorphisms $\mathrm{Pol}(\Gamma)$ of a relational structure forms an algebraic object called *clone* [28], which is a set of operations defined on a set D that is closed under composition and that contains all projections. Moreover, $\mathrm{Pol}(\Gamma)$ is also closed under interpolation: We say that a k-ary operation f is *interpolated* by a set of k-ary operations F if for every finite subset A of D there is some operation $g \in F$ such that $f(a) = g(a)$ for every $a \in A^k$. A clone that contains all operations that are interpolated by operations in the clone is called *locally closed*. We say that a set of operations F *locally generates* an operation g if g is in the smallest locally closed clone containing all operations in F.

Polymorphism clones can be used to characterize primitive positive definability in finite structures, by a result of [6] and [19]. In general, this is not true for infinite-domain constraint languages. However, the result remains true if the relational structures are ω-categorical.

Theorem 6. *[5] Let Γ be an ω-categorical constraint language. Then the relations preserved by the polymorphisms of Γ are precisely those that have a primitive positive definition in Γ.*

Lemma 7. *Let Γ be a ω-categorical constraint language and let R be a k-ary relation that is a union of l orbits of k-tuples of $Aut(\Gamma)$. If R has no primitive positive definition in Γ, then Γ has an at most l-ary polymorphism violating R.*

5. POLYMORPHISM CLONES OF TEMPORAL CONSTRAINT LANGUAGES

In this paper, we always deal with polymorphism clones of temporal constraint languages, and it is convenient to make the following convention. We say that a set of operations F *generates* an operation g if F together with all automorphisms of $(\mathbb{Q}, <)$ locally generates g. In case that F contains just one operation f, we also say that f generates g.

The polymorphism clone of a temporal constraint language generated by a single operation can be described differently as follows. A k-ary operation f on $\mathbb{Q}$ defines a weak linear order $\preceq$ on $\mathbb{Q}^k$, as follows: for $x, y \in \mathbb{Q}^k$, let $x \preceq y$ iff $f(x) \le f(y)$. The following observation easily follows from the properties of $\mathrm{Aut}(\mathbb{Q}, <)$ and local closure.

Observation 1. *Let f and g be two k-ary operations that define the same weak linear order on $\mathbb{Q}^k$. Then f generates g and g generates f.*

We now define fundamental operations on $\mathbb{Q}$. The unary operation $-$ is defined as $-(x) := -x$ in the usual sense, and the unary operation *cyc* is defined as $-1/(x-1)$ for $x \le 0$ and $-1/(x+1)$ for $x > 0$. Cameron's theorem can now be reformulated (and slightly extended) in terms of endomorphisms of temporal constraint languages.

Proposition 8. *Let Γ be a temporal constraint language. Then one of the following cases applies.*

- *Γ has a constant endomorphism;*
- *All endomorphisms of Γ preserve $<$;*
- *The set of endomorphisms of Γ equals the set of unary operations generated by $-$;*
- *The set of endomorphisms of Γ equals the set of unary operations generated by cyc;*
- *The set of endomorphisms of Γ equals set of unary operations generated by $-$ and cyc;*
- *The set of endomorphisms of Γ equals the set of all injective unary operations.*

Let f be a k-ary operation on $\mathbb{Q}$. The operation $-f(-x_1, \ldots, -x_k)$ is called the *dual* of f. Note that if f preserves an m-ary relation R, then the dual of f preserves the relation $-R$, which is defined as $\{(-t_1, \ldots, -t_m) \mid (t_1, \ldots, t_m) \in R\}$. Clearly, if the CSP of a temporal constraint language $(\mathbb{Q}, R_1, R_2, \ldots)$ can be solved in polynomial time (is NP-complete), then also the CSP of $(\mathbb{Q}, -R_1, -R_2, \ldots)$ can be solved in polynomial time (is NP-complete, respectively).

6. SOME HARD TEMPORAL CSPS

We have already mentioned in the introduction that the Betweenness and the Cyclic Ordering problem in [18] can be formulated as temporal CSPs, and that these problems are NP-complete. The corresponding relations *Betw* and *Cycl* re-appeared in Cameron's theorem. The separation relation *Sep* is another relation that appeared in Theorem 2, and the corresponding CSP is again NP-complete.

Proposition 9. *The problem CSP(Sep) is NP-complete.*

An important relation for our classification is the relation S, defined as follows.

Definition 2. *Let S be the ternary relation $\{(x, y, z) \in \mathbb{Q}^3 \mid x = y < z \vee x = z < y\}$.*

Proposition 10. *The problem CSP(S) is NP-complete.*

We would like to remark that our classification result (Theorem 27) implies that if there is no primitive positive definition of *Betw*, *Cycl*, *Sep*, S, $-S$, or $\{(x, y, z) \in \mathbb{Q}^3 \mid x = y \neq z \vee x \neq y = z\}$ in Γ, then Γ is tractable.

7. LANGUAGES PRESERVED BY PP

Let pp be a binary operation on $\mathbb{Q}$ such that $pp(a, b) \leq pp(a', b')$ if ($a \leq 0$ and $a \leq a'$) or ($a > 0$, $a' > 0$, and $b \leq b'$). For an illustration of such an operation and its dual, see Figure 1. In diagrams for binary operations f as in Figure 1, we draw a directed edge from (a, b) to (a', b') if $f(a, b) < f(a', b')$. Unoriented lines in the picture relate pairs of values that have the same value under f. By Observation 1, all such operations generate the same clone and thus it does not matter which operation satisfying the above conditions we pick. The same argument also applies to all other definitions of operations in this paper and we thus do not repeat it again.

It is easy to verify that the relation S, defined in Section 6, is preserved by pp. Proposition 10 shows that CSP(S) is NP-complete. In our quest for tractable constraint languages, we therefore study further restrictions of temporal constraint languages.

Definition 3. *Let f be a binary operation preserving $<$. If $f(0, 0) = f(0, x) = f(x, 0)$ for all integers $x > 0$, we say that f* provides min-union closure. *We say that f* provides min-intersection closure *if $f(0, 0) < f(0, x)$ and $f(0, 0) < f(x, 0)$ for all integers $x > 0$. We say that f* provides min-xor closure *if $f(0, 0) > f(0, x) = f(x, 0)$ for all integers $x > 0$.*

The binary operation *min* that maps its two arguments to the minimum of the two arguments is an example of an operation that generates pp and provides min-union closure (see Figure 2). The following proposition implies that $\{f, pp\}$ generates *min* for every operation f that provides min-union closure.

Proposition 11. *A temporal relation is preserved by pp and an operation providing min-union closure if and only if it is preserved by min.*

For constraint languages over a finite domain, *min*- and *max*-closed relations were studied in [25]. An equivalent clausal description of such constraints is known; however, the equivalence only holds for *finite* domains. The tractability of the clausal description of *max* and *min*-closed constraints has also been shown for infinite domains [14]. But, unlike our algorithm, the algorithm presented in [14] cannot be applied to *min*-closed constraint languages over an infinite domain.

We now define an operation, called *mi*, that clearly generates *pp* and provides min-intersection closure.

$$mi(x, y) := \begin{cases} \alpha(x) & \text{if } x < y \\ \beta(x) & \text{if } x = y \\ \gamma(y) & \text{if } x > y \end{cases}$$

where α, $\beta, \gamma \in Aut(\mathbb{Q}, <)$ such that $\beta(x) < \gamma(x) < \alpha(x) < \beta(x + \varepsilon)$ for all $x \in \mathbb{Q}$ and all $0 < \varepsilon \in \mathbb{Q}$ (see Figure 2). In fact, the operation *mi* will be of special importance, because the following proposition shows that *pp* together with any operation providing min-intersection closure generates the operation *mi*.

Proposition 12. *A temporal relation R is preserved by pp and a polymorphism f providing min-intersection closure if and only if it is preserved by mi.*

An example of a relation that is preserved by *mi* but not by *min* is the (unique) smallest temporal relation I containing the tuples $(0, 0, 1, 2), (0, 0, 1, 1), (1, 1, 0, 0), (2, 1, 0, 0)$. An example of a ternary relation that is preserved by *min* but not by *mi* is the relation $U(x, y, z)$ defined by $(x = y \wedge y < z) \vee (x = z \wedge z < y) \vee (x = y \wedge y = z)$.

We now define an operation, called *mx*, that clearly generates *pp* and provides min-xor closure.

$$mx(x, y) := \begin{cases} \alpha(min(x, y)) & \text{if } x \neq y \\ \beta(x) & \text{if } x = y \end{cases}$$

where α and β are automorphisms of $(\mathbb{Q}, <)$ such that $\alpha(x) < \beta(x) < \alpha(x+\varepsilon)$ for all $x \in \mathbb{Q}$ and all $0 < \varepsilon \in \mathbb{Q}$ (see Figure 2; note that in the picture for *mx*, the value for $f(x, y)$ equals $f(y, x)$). The following proposition implies that $\{f, pp\}$ generates *mx* for any operation f providing min-xor closure.

Proposition 13. *A temporal relation R is preserved by pp and an operation f providing min-xor closure if and only if R is preserved by mx.*

An example of a temporal relation that is preserved by *mx* (but not by *min* and by *mi*) is the ternary relation $X(x, y, z)$ defined by $(x = y \wedge y < z) \vee (x = z \wedge z < y) \vee (y = z \wedge y < x)$.

8. ALGORITHMS FOR LANGUAGES PRESERVED BY PP

We now describe algorithms for constraint languages that are preserved by *min*, *mi*, or *mx*. They all follow a common strategy: they are searching for a variable that can have the minimal value in a solution. If the algorithms have found such a variable, say x, the algorithms add equalities and inequalities that are implied by all constraints under the assumption that x denotes the minimal value in *all* solutions. Next, the algorithms recursively solve the instance consisting of the projections of all constraints to the variables that do not denote the minimal value in all solutions. We show that for languages preserved by *pp* it is true that if the instance has a solution, it also has a solution that satisfies all the additional constraints.

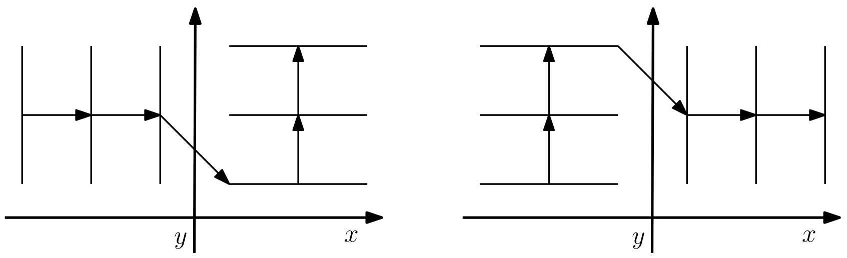

Figure 1: A visualization of *pp* (left) and *dual-pp* (right).

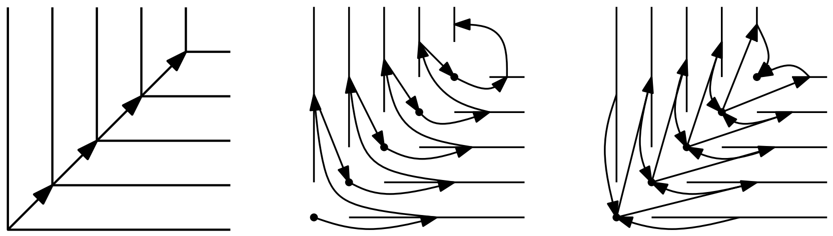

Figure 2: Illustration of the operation *min* (left), *mi* (center), and *mx* (right).

DEFINITION 4. *Let R be an n-ary temporal relation and $S = \{p_1, \ldots, p_k\}$, for $1 \le p_1 < \cdots < p_k \le n$, a subset of indices. Let $\{q_1, \ldots, q_l\}$ be $[n] \setminus S$. Then the* ordered projection *of R to S is the relation $R'(x_{p_1}, \ldots, x_{p_k})$ with the primitive positive definition*

$$\exists x_{q_1}, \ldots, x_{q_l}. R(x_1, \ldots, x_n) \wedge \bigwedge_{i \in S, j \in [n] \setminus S} x_i < x_j \,.$$

Note that there are only finitely many ordered projections of relations in Γ. In case that there is a primitive positive definition of $<$ in Γ, ordered projections are primitive positive definable. By Lemma 5, we can assume in this case that Γ contains all relations defined by ordered projections. To formally introduce our algorithms, we also need the concept of an ordered projection of *instances* of the CSP.

DEFINITION 5. *Let Φ be an instance of $CSP(\Gamma)$ and $X \subset V(\Phi)$. The* ordered projection of Φ to X *is an instance Φ' of $CSP(\Gamma)$ defined as follows. For each constraint $R(x_1, \ldots, x_n)$ in Φ with $\{x_1, \ldots, x_n\} \cap X = \{x_{k_1}, \ldots, x_{k_l}\}$, $k_1 < \cdots < k_l$, the instance Φ' contains the constraint $R'(x_{k_1}, \ldots, x_{k_l})$ where R' is the ordered projection of R to $\{k_1, \ldots, k_l\}$.*

DEFINITION 6. *Let Φ be an instance of a temporal CSP and let t be from $\mathbb{Q}^k$. The i-th entry in t is called* minimal *if $t[i] \le t[j]$ for every $j \in [k]$. The set of entries that are minimal in t is called the min-set (of t), and denoted by $M(t)$. If $\psi = R(x_1, \ldots, x_k)$ is a constraint from Φ, then a subset $\{x_{k_1}, \ldots, x_{k_s}\}$ of the variables of ψ is called a* min-set *(of ψ) if there exists a k-tuple t satisfying ψ such that $t[k_1], \ldots, t[k_s]$ are the minimal entries in t (assuming that $k_1 < \cdots < k_s$). A set of variables $S \subset V(\Phi)$ is called* free *iff for all constraints $R(x_1, \ldots, x_k)$ in Φ the set $S \cap \{x_1, \ldots, x_k\}$ is either empty or a min-set of R.*

We will show how to use the concept of freeness to solve instances Φ of CSP(Γ) for languages Γ preserved by *pp*.

LEMMA 14. *Let Φ be an instance of $CSP(\Gamma)$ where Γ is preserved by pp, and let S be a free set of variables of Φ. Then Φ has a solution if and only if the ordered projection Φ' of Φ to $V(\Phi) \setminus S$ has a solution.*

PROOF. First suppose Φ' has a solution s'. Let $\psi = R(x_1, \ldots, x_m)$ be a constraint of Φ such that $V(\psi) \cap S = \{x_{p_1}, \ldots, x_{p_k}\} \neq \emptyset$. Let $\{x_{q_1}, \ldots, x_{q_l}\} = V(\psi) \setminus S$ for $q_1 < \cdots < q_l$. By the definition of an ordered projection, there is a tuple $t_1 \in R$ such that $s'(x_i) = t_1[i]$ for all $i \in \{q_1, \ldots, q_l\}$. Since $V(\psi) \cap S$ is a min-set of R, there is a tuple $t_2 \in R$ such that $M(t_2) = \{p_1, \ldots, p_k\}$. Let $\alpha \in Aut(\mathbb{Q}, <)$ be such that α maps the minimal value of t_2 to 0. Because R is preserved by *pp*, the tuple $t_3 := pp(\alpha(t_2), t_1)$ is in R. It is easy to verify that $M(t_3) = \{p_1, \ldots, p_k\}$ and that there is $\beta \in Aut(\mathbb{Q}, <)$ such that $\beta(t_3[i]) = s'(x_i)$ for $i \in \{q_1, \ldots, q_l\}$. Because we can find such a tuple for all the constraints ψ in Φ where $V(\psi) \cap S \neq \emptyset$, we conclude that a solution s' of Φ' can be extended to a solution s of Φ by setting all the variables in S to some value that is smaller than the smallest value in $\{s'(x) \mid x \in V(\Phi')\}$. Clearly, all the constraints ψ in Φ with $V(\psi) \cap S = \emptyset$ or $V(\psi) \subset S$ are satisfied by s as well.

Now suppose that Φ has a solution s. Let $x_1, \ldots, x_n$ be the variables of Φ, and let $\{x_{r_1}, \ldots, x_{r_{|S|}}\}$ be S. Let s' be a mapping from $V(\Phi)$ to $\mathbb{Q}$ such that $M((s'(x_1), \ldots, s'(x_n))) = \{r_1, \ldots, r_{|S|}\}$, and $s'(x) = s(x)$ for $x \in V(\Phi) \setminus S$. We claim that s' is a solution for Φ. Let $\psi = R(y_1, \ldots, y_m)$ be a constraint of Φ such that $V(\psi) \cap S \neq \emptyset$. Clearly, $t_1 := (s(y_1), \ldots, s(y_m))$ is in R since s is a solution of Φ. Let $\{y_{p_1}, \ldots, y_{p_l}\}$ be $S \cap \{y_1, \ldots, y_m\}$. Since $\{y_{p_1}, \ldots, y_{p_l}\}$ is a min-set of R, there is a tuple $t_2 \in R$ such that $M(t_2) = \{p_1, \ldots, p_l\}$. Let $\alpha \in Aut(\mathbb{Q}, <)$ be such that α maps the minimal value of t_2 to 0. Because R is preserved by *pp*, the tuple $t_3 := pp(\alpha(t_2), t_1)$ is in R. It is easy to verify that $M(t_3) = \{p_1, \ldots, p_l\}$, and that there is an automorphism β such that $\beta(t_3)[i] = s(y_i)$ for $i \in [m] \setminus \{p_1, \ldots, p_l\}$. Clearly, the restriction of s' to $V(\Phi) \setminus S$ is a solution to the ordered projection Φ' of Φ to $V(\Phi) \setminus S$ since s' also satisfies all the inequalities imposed by the ordered projection. Therefore Φ' is satisfied by s'. □

The above lemma implies that if Γ is preserved by *pp* and if we are able to identify a free set for instances of CSP(Γ) in polynomial time, then we also have a polynomial time algorithm that solves CSP(Γ). The running time of the algorithm is $O(n \cdot (m + t(n, m)))$, where $n = |V|$, m is the number of constraints in Φ, and $t(n, m)$ is the running time of the procedure that computes the free set in terms of n and m.

We now study how to find free sets in constraint languages Γ that are preserved by *min*. Let $\psi = R(x_1, \ldots, x_k)$ be a constraint where R is from Γ, and let L be a subset of $\{x_1, \ldots, x_k\}$. Let $A_1, \ldots, A_l$ be all min-sets of ψ such that $A_i \subseteq L$ for all $i \in [l]$. If $l \geq 1$, i.e., if such a min-set exists, there is a unique min-set A_j, $j \in [l]$, with the property that $A_i \subseteq A_j$ for all $i \in [l]$, because R is preserved by *min*. We call A_j the *maximal min-set of ψ contained in L*.

The algorithm for finding a free set starts with the set of all variables as a candidate set S. For each constraint ψ the algorithm determines the maximal min-set M of ψ contained in S and replaces S by $S \cap M$. This step is repeated until no variable is removed from S. It can be shown that if the resulting set is empty, then Φ does not have a solution, and that otherwise we have found a free set.

THEOREM 15. *If Γ is preserved by min there is an algorithm solving CSP(Γ) in time $O(n^2 m)$.*

Next, we study instances Φ of CSP(Γ) where Γ is preserved by *mi*. Let $\psi = R(x_1, \ldots, x_k)$ be a constraint in Φ, and let $L \subseteq \{x_1, \ldots, x_k\}$. Let $A_1, \ldots, A_l$ be all min-sets of ψ such that $L \subseteq A_i$ for all $i \in [l]$. It can be verified that there is a min-set A_j of R that is a subset of every min-set containing L. We call A_j the *minimal min-set of R containing L*.

The algorithm for finding a free set starts with $S := \{x\}$ for some variable x. In each step the algorithm determines for each constraint ψ in Φ the minimal min-set M of ψ containing S. If there is no such min-set, the algorithm proceeds with another variable x. Otherwise, S is replaced by $S \cup M$, and the algorithm proceeds with another constraint, until no variable is added to S. It can be shown that in this case the resulting set is free, and that Φ does not have a solution if this procedure fails to find a free set for all variables x.

THEOREM 16. *If Γ is preserved by mi there is an algorithm solving CSP(Γ) in time $O(n^3 m)$.*

Finally, we consider languages Γ preserved by *mx*. Let $R(x_1, \ldots, x_k)$ be a constraint with R from Γ. For a tuple $t \in R$, we define $\chi_{min}(t)$ to be a vector from $\{0,1\}^k$ such that $\chi_{min}(t)[i] = 1$ if and only if $t[i]$ is minimal in t. We define $\chi_{min}(R)$ to be $\{\chi_{min}(t) \mid t \in R\}$. Since R is preserved by *mx*, the set $\chi_{min}(R)$ is closed under addition of distinct vectors over $GF(2)$ and so $\chi_{min}(R) \cup \{0^k\}$ is also closed under the operation $g(x, y, z) := x+y+z$ over $GF(2)$. Such an operations is called *affine* (usually, affine operations are defined as $x + y - z$, but for $GF(2)$ this is equivalent) and a set of vectors preserved by the affine operation is exactly the set of solutions of a system of linear equations; see e.g. [15]. Moreover, this system can be constructed in polynomial time. So if we have an instance Φ of CSP(Γ), we can find a free set of variables as follows (if it exists): we first construct a system of linear equations over $GF(2)$ with the set of variables $x_v, v \in V$, and equations as described above for each constraint in Φ. Then we compute a solution of the system that is distinct from 0^k (essentially by Gaussian elimination). If there is such a solution, then the set of variables mapped to 1 is a free set of Φ. If S has no such solution, then there is no free set of variables, and there is no solution for Φ.

THEOREM 17. *If Γ is preserved by mx there is an algorithm solving CSP(Γ) in time $O(n(m + n^3))$.*

We are now ready to classify the complexity of the temporal constraint languages that are preserved by *pp*.

LEMMA 18. *Let f be a binary operation that preserves $<$ and violates the relation S. Then $\{f, pp\}$ generates an operation providing min-intersection, min-union, or min-xor closure.*

THEOREM 19. *Let Γ be a temporal constraint language such that $<$ has a pp-definition in Γ and $pp \in Pol(\Gamma)$. If S has a pp-definition in Γ, then CSP(Γ) is NP-complete. Otherwise Γ is tractable.*

PROOF. If S has a primitive positive definition in Γ, then Proposition 10 shows that CSP(Γ) is NP-hard. Otherwise, Lemma 7 implies that there is an at most binary polymorphism f of Γ that violates S (and preserves $<$ as $<$ has a pp-definition in Γ). We can therefore apply Lemma 18 and find a binary polymorphism h of Γ providing min-intersection, min-union, or min-xor closure. In each of these cases, because Γ is preserved by *pp*, Proposition 11, 12, or 13 assert that Γ is preserved under *min*, *mi*, or *mx* and thus we can conclude using Theorem 15, 16, or 17 that Γ is tractable. □

9. THE FULL CLASSIFICATION

Temporal constraint languages where not all first-order definable relations have a primitive positive definition can be divided into three (overlapping) groups: those preserved by *pp*, those preserved by *dual-pp*, and those preserved by an operation called *lex*. In this section we focus on languages preserved by *lex* and complete the classification. Operation *lex* is a binary operation on $\mathbb{Q}$ such that $lex(a, b) < lex(a', b')$ if either $a < a'$, or $a = a'$ and $b < b'$. Note that every operation *lex* satisfying these conditions is by definition injective. It is easy to see that the relation *Betw* is preserved by *lex*. Therefore, we are interested in further restrictions of languages preserved by *lex* that imply tractability of the corresponding CSP. One restriction and its dual counterpart are introduced in [4]; again, these restrictions are defined in terms of binary polymorphisms.

Let *ll* be a binary operation on $\mathbb{Q}$ such that $ll(a, b) < ll(a', b')$ if ($a \leq 0$ and $a < a'$) or ($a \leq 0$ and $a = a'$ and $b < b'$) or ($a, a' > 0$ and $b < b'$) or ($a > 0$ and $b = b'$ and $a < a'$). All operations satisfying these conditions are by definition injective, and they all generate the same clone. For an illustration of *ll* and its dual, see Figure 3. The class of Ord-Horn relations mentioned in the introduction is preserved by these operations. It is easy to see that *ll* generates *lex*.

To talk about properties of an operation on restricted subsets of the domain $\mathbb{Q}$, the following concepts are useful: If

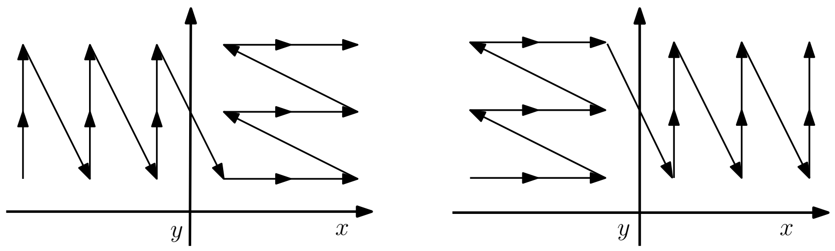

Figure 3: A visualization of *ll* (left) and *dual-ll* (right).

$S_1, \ldots, S_d$ are sets, we call a set of the form $S_1 \times \cdots \times S_d$ a *grid*, and also write S^d for a product of the form $S \times \cdots \times S$ with d factors. A $[k]^d$*-subgrid* of a grid $S_1 \times \cdots \times S_d$ is a subset of $S_1 \times \cdots \times S_d$ of the form $S'_1 \times \cdots \times S'_d$, where S'_i is a k-element subset of S_i. We say that a *k-ary operation f behaves like a k-ary operation g on $G \subset \mathbb{Q}^k$* if the weak linear order induced by f on the tuples in G is the same as the order induced on these tuples by g.

The following two technical lemmas are used in the proof of Lemma 23:

LEMMA 20. *Let f be a binary operation violating Betw and preserving $<$. Then there are $t_1, t_2 \in$ Betw such that $f(t_1, t_2)$ has three distinct entries and $f(t_1, t_2) \notin$ Betw.*

LEMMA 21. *Let f be a binary operation, let $c, d \in \mathbb{Q}$, and let $S^{(i)}, T_1^{(i)}, T_2^{(i)} \subseteq \mathbb{Q}$ for every $i > 0$ be such that*

- $|S^{(i)}| = |T_1^{(i)}| = |T_2^{(i)}| \geq i$,
- $t_1 < d < t_2$ *for all* $t_1 \in T_1^{(i)}$, $t_2 \in T_2^{(i)}$,
- $f(s, t_1) < c < f(s, t_2)$ *for all* $s \in S^{(i)}, t_1 \in T_1^{(i)}, t_2 \in T_2^{(i)}$,
- *f behaves like a projection to the first argument on* $S^{(i)} \times T_1^{(i)}$ *and on* $S^{(i)} \times T_2^{(i)}$.

(See Figure 4 for an illustration.) Then f generates lex(y, x).

The next lemma (Lemma 23) demonstrates how we use the Product Ramsey theorem in our classification proof. The product Ramsey theorem can be derived easily from the classical infinite Ramsey theorem; see [26] for a general introduction to Ramsey theory.

THEOREM 22. *For all positive integers d, r, m, and k, there is a positive integer $\mathcal{R}(d, r, m, k)$ such that for all sets $S_1, \ldots, S_d$, $|S_i| \geq \mathcal{R}(d, r, m, k)$ for all $i \in [d]$, and an arbitrary coloring of the $[m]^d$ subgrids of $S_1 \times \ldots \times S_d$ with r colors, there exists a $[k]^d$ subgrid of $S_1 \times \ldots \times S_d$ such that all $[m]^d$ subgrids of the $[k]^d$ subgrid have the same color.*

The PRT is also used in the proof of Lemma 25 in a similar way as in the proof of Lemma 23.

LEMMA 23. *Let f be a binary operation that preserves $<$ and violates Betw. Then f generates lex, pp, or dual-pp.*

PROOF. As f violates *Betw* and preserves $<$, Lemma 20 asserts that there are $t_1, t_2 \in$ *Betw* such that $t := f(t_1, t_2) \notin$ *Betw* and t has pairwise distinct entries. As f preserves $<$, we can assume without loss of generality that $t_1[1] < t_1[2] < t_1[3]$ and $t_2[1] > t_2[2] > t_2[3]$ (otherwise, we apply the argument to $f(y, x)$).

Either the triple t satisfies $t[1] > t[2] < t[3]$ or $t[1] < t[2] > t[3]$. In the first case, let $S_1 := \{x \in \mathbb{Q} \mid t_1[1] < x < t_1[2]\}$, $S_2 := \{x \in \mathbb{Q} \mid t_1[3] < x\}$, $T_1 := \{y \in \mathbb{Q} \mid t_2[3] < y < t_2[2]\}$, and $T_2 := \{y \in \mathbb{Q} \mid t_2[1] < y\}$. In the second case, let $S_1 := \{x \in \mathbb{Q} \mid t_1[2] < x < t_1[3]\}$, $S_2 := \{x \in \mathbb{Q} \mid x < t_1[1]\}$, $T_1 := \{y \in \mathbb{Q} \mid t_2[2] < y < t_2[1]\}$, and $T_2 := \{y \in \mathbb{Q} \mid y < t_2[3]\}$. See Figure 5 for an illustration of these sets.

We apply the PRT for $d = m = 2$ and a sufficiently large integer k to the grid $S_1 \times T_1$. We color each $[2] \times [2]$ subgrid of this grid according to the weak linear order of the four elements (see Figure 6). As S_1 and T_1 are infinite and, in particular, larger than $\mathcal{R}(\mathcal{R}(k))$ (we ommit d, r and m from the notation since they are fixed), the PRT asserts that there are subsets $U^{(k)} \subseteq S_1$ and $V^{(k)} \subseteq T_1$ such that $|U^{(k)}| \geq \mathcal{R}(k)$, $|V^{(k)}| \geq \mathcal{R}(k)$, and the $[2] \times [2]$ subgrids of $U^{(k)} \times V^{(k)}$ are monochromatic. Next, we similarly apply the PRT to the grid $U^{(k)} \times T_2$ for each $k > 0$ and obtain subsets $S_1^{(k)} \subseteq U^{(k)}$ and $T_2^{(k)} \subseteq T_2$. We finally apply the PRT to the grid $S_2 \times V^{(k)}$ for each $k > 0$ and obtain subsets $S_2^{(k)} \subseteq S_2$ and $T_1^{(k)} \subseteq V^{(k)}$. By construction, grids $S_1^{(k)} \times T_2^{(k)}$ and $S_2^{(k)} \times T_1^{(k)}$ have the same linear order induced on all $[2] \times [2]$ subgrids, $|S_1^{(k)}| \geq k$, $|S_2^{(k)}| \geq k$, $|T_1^{(k)}| \geq k$, and $|T_2^{(k)}| \geq k$. The only linear order that can be induced on $[2] \times [2]$ subgrids of a large grid are the first two order depicted in the first line of Figure 6 and the first four order depicted in the second line of Figure 6. Therefore, on each of those three grids f behaves either as a projection to one of arguments, as *lex*(x, y), as *lex*(y, x), as *lex*$(x, -y)$, or as *lex*$(y, -x)$. We see that f can behave on each of the grids in 6 different ways and thus there are just 6^3 possibilities how f behaves on the grids for given k. Hence, there is an infinite set $K \subseteq \mathbb{N}$ such that f behaves the same way on the grids $S_1^{(k)} \times T_1^{(k)}$ for all $k \in K$, the same way on the grids $S_1^{(k)} \times T_2^{(k)}$ for all $k \in K$, and the same way on the grids $S_2^{(k)} \times T_1^{(k)}$ for all $k \in K$.

If f behaves like *lex*(x, y), *lex*(y, x), *lex*$(x, -y)$, or *lex*$(y, -x)$ on some of the grids for all $k \in K$, it clearly generates either *lex*(x, y) or *lex*$(x, -y)$. Since *lex*$(x, -$*lex*$(x, -y))$ induces the same order on $\mathbb{Q}^2$ as *lex*, the operation *lex*$(x, -y)$ generates *lex*, and thus we are done.

So assume that f behaves like a projection on all three grids for all $k \in K$. Observe that by the choice of $S_1^{(k)}$, $S_2^{(k)}$, $T_1^{(k)}$, and $T_2^{(k)}$, and because f preserves $<$, we have that either $f(x, y) < f(t_1[2], t_2[2]) < f(x', y')$ for all $(x, y) \in S_1^{(k)} \times T_1^{(k)}$, $(x', y') \in (S_1^{(k)} \times T_2^{(k)}) \cup (S_2^{(k)} \times T_1^{(k)})$ (this

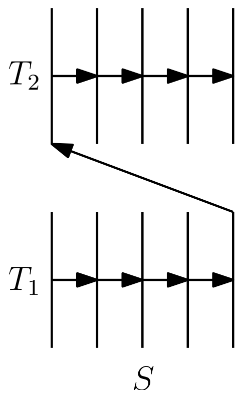

Figure 4: An operation from Lemma 21.

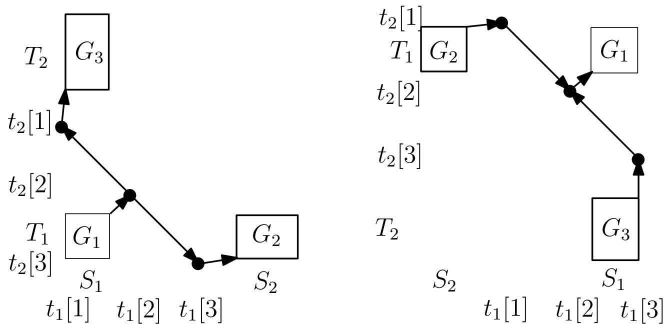

Figure 5: Grids chosen for the application of the product Ramsey theorem. The depicted order on the values of f follows from the choice of t_1, t_2 and because f preserves $<$.

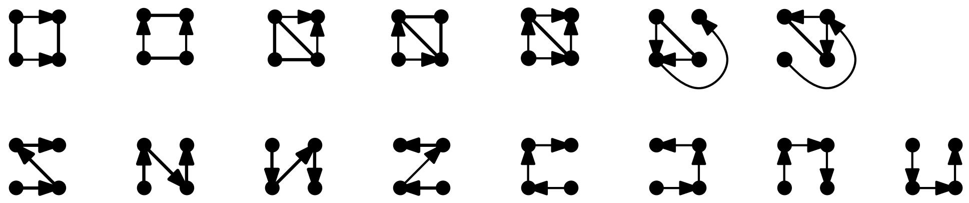

Figure 6: Possible weak linear orders on $[2] \times [2]$ grids.

is in case $S_1^{(k)}$ is below $S_2^{(k)}$ and $T_1^{(k)}$ is below $T_2^{(k)}$), or $f(x,y) > f(t_1[2], t_2[2]) > f(x',y')$ for all $(x,y) \in S_1^{(k)} \times T_1^{(k)}$, $(x',y') \in (S_1^{(k)} \times T_2^{(k)}) \cup (S_2^{(k)} \times T_1^{(k)})$ (this is in case $S_1^{(k)}$ is above $S_2^{(k)}$ and $T_1^{(k)}$ is above $T_2^{(k)}$). Now, we distinguish two cases. Either f behaves like a projection to the same argument on each of the three grids for all $k \in K$ or f behaves like a projection to the first argument on one grid and as a projection to the second argument on some other grid for all $k \in K$. In the first case, suppose without loss of generality that f behaves on each grid as a projection to the first argument (otherwise we can swap arguments). Due to the observation from the beginning of the paragraph, we can apply Lemma 21 and conclude that f generates *lex*.

Now, we turn our attention to the second case and show that f generates *pp* or *dual-pp*. Again, we assume without loss of generality that f behaves like a projection to the first argument on $S_1^{(k)} \times T_1^{(k)}$ for all $k \in K$. Let $S_1^{(k)} \times T_2^{(k)}$ be the grid where f behaves like a projection to the second argument (the other case is symmetric). Due to the observation from the beginning of the previous paragraph we immediately see that f behaves either as *pp* (this is in case $T_1^{(k)}$ is above $T_2^{(k)}$) or as *dual-pp* (this is in case $T_1^{(k)}$ is below $T_2^{(k)}$) on the grid $S_1^{(k)} \times (T_1^{(k)} \cup T_2^{(k)})$ after swapping arguments for all $k \in K$. Thus f generates *pp* or *dual-pp*. □

LEMMA 24. *Let f be a binary operation that preserves $<$ and violates Betw. Then $\{f, lex\}$ generates a binary injective operation that preserves $<$ and violates Betw.*

LEMMA 25. *Let f be a binary injective operation preserving $<$ and violating Betw. Then f generates ll or dual-ll.*

In the proof of the following theorem we make essential use of Cameron's theorem:

THEOREM 26. *Let Γ be a temporal constraint language. Then it satisfies at least one of the following:*

1. *There is a primitive positive definition of Cycl, Betw, or Sep in Γ.*
2. *Pol(Γ) contains a constant operation.*
3. *Aut(Γ) contains all permutations of $\mathbb{Q}$.*
4. *There is a primitive positive definition of $<$ in Γ and Pol(Γ) contains a binary operation f violating Betw.*

PROOF. If there is a primitive positive definition of *Betw* in Γ we are in the first case and done. Otherwise, Lemma 7 shows that there is a binary polymorphism of Γ that violates *Betw*. If there is a primitive positive definition of $<$ in Γ, we are in case four and done. Otherwise, again by Lemma 7, there is a unary polymorphism of Γ that violates $<$. Proposition 8 shows that Γ is preserved by a constant, $-$, or *cyc*. In the first case we are done, so we assume in the following that Γ is not preserved by a constant.

In the second case, the relation *Betw* consists of only one orbit of triples, and Lemma 7 shows that there is an endomorphism that violates *Betw*. Proposition 8 then implies that Γ is preserved by *cyc* (and by $-$). In this case, the relation *Sep* consists of only one orbit of 4-tuples. Again, either *Sep* has a primitive positive definition, and the first item of the theorem is true, or there is an endomorphism that violates *Sep*. Proposition 8 now shows that Γ is preserved by all injective unary operations.

In the third case, the relation *Cycl* consists of only one orbit of triples. If *Cycl* has a primitive positive definition in Γ, the first item of the theorem is true. Otherwise, Lemma 7 shows that there is an endomorphism that violates *Cycl*. Proposition 8 then shows that Γ is preserved by $-$ (and by *cyc*). This case we have already discussed in the last paragraph. □

We are now ready to state and prove our classification result.

THEOREM 27. *A temporal constraint language Γ is tractable if Γ is preserved by at least one of the following nine operations: ll, min, mi, mx, their duals, or a constant operation. Otherwise, CSP(Γ) is NP-complete.*

PROOF. If Γ is preserved by a constant operation, then assigning the same value to every variable of an instance Φ of CSP(Γ) is a solution of I, unless there is a constraint for false in the instance, in which case we simply reject. Thus CSP(Γ) is clearly tractable. In case that Γ is preserved by *ll* or *dual-ll*, there is a quadratic time algorithm that solves CSP(Γ), see [4]. If Γ is preserved by *min*, *mi*, *mx* or one of their duals, tractability is shown in Section 7.

Theorem 26 asserts that one of the following cases it true: CSP(Γ) is NP-complete (because there is a primitive positive definition of some relation with an NP-complete CSP), *Pol*(Γ) contains a constant operation (in which case CSP(Γ) is tractable as we have argued above), *Pol*(Γ) contains all permutations of $\mathbb{Q}$, or there is a primitive positive definition of $<$ in Γ and a binary $f \in Pol(\Gamma)$ that violates *Betw*. In the third case, Γ is an equality constraint language, and the statement follows easily from [3]. In the fourth case, Lemma 23 implies that the operation f generates *pp*, *dual-pp*, or *lex*. If Γ is preserved by *pp* or *dual-pp*, we are in one of the described tractable cases or NP-complete by Theorem 19 and its dual counterpart. If Γ is preserved by *lex*, then by Lemma 24 the operation f generates a binary *injective* operation g preserving $<$ and violating *Betw*. Finally, using Lemma 25 we get that g generates *ll* or *dual-ll* and thus we are in one of the described tractable cases. □

10. CONCLUDING REMARKS

See Table 1 for an overview over the nine largest tractable temporal constraint languages; the entries also mention *typical relations* for the respective language, i.e., a set of relations that is contained in the language, but not contained in any other of the nine languages – hence, these relations show that all the languages are distinct. Note that from the description of the constraint languages via polymorphisms it also follows that the so-called *meta-problem* for tractability is decidable, i.e., there is an algorithm that decides for a given finite temporal constraint language (given e.g. by defining first-order formulas for the relations in the language) whether it is tractable or has an NP-complete CSP.

11. REFERENCES

[1] M. Bodirsky. Cores of countably categorical structures. *Logical Methods in Computer Science (LMCS)*, 2007. DOI: 10.2168/LMCS-3(1:2).

[2] M. Bodirsky and V. Dalmau. Datalog and constraint satisfaction with infinite templates. In *Proceedings of the 23rd International Symposium on Theoretical*

Number	Polymorphism	Typical Relations	Complexity of Algorithm
1	min	$\{U, <\}$	$O(n^2m)$
2	mi	$\{I\}$	$O(n^3m)$
3	mx	$\{X\}$	$O(n(m+n^3))$
4	max = dual min	$\{-U, <\}$	$O(n^2m)$
5	dual mi	$\{-I\}$	$O(n^3m)$
6	dual mx	$\{-X\}$	$O(n(m+n^3))$
7	ll	$\{(u \neq v) \vee x > y \vee x > z\}$	$O(nm)$
8	dual ll	$\{(u \neq v) \vee x < y \vee x < z\}$	$O(nm)$
9	constant	$\{(x \leq y \leq z) \vee (z \leq y \leq z)\}$	$O(m)$

Table 1: Summary of the nine tractable languages. For the last three languages, the typical relations are just given by their first-order definition; in all other cases, see Section 7.

Aspects of Computer Science (STACS'06), LNCS 3884, pages 646–659. Springer Verlag, 2006.

[3] M. Bodirsky and J. Kára. The complexity of equality constraint languages. *Accepted for publication in Theory of Computing Systems*, 2007. A conference version of the paper appeared in the proceedings of the International Computer Science Symposium in Russia (CSR'06).

[4] M. Bodirsky and J. Kára. A fast algorithm and lower bound for temporal reasoning. Preprint, 2007.

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

[6] 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.

[7] A. Bulatov. A dichotomy theorem for constraints on a three-element set. In *FOCS'02*, pages 649–658, 2002.

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

[9] A. Bulatov. A graph of a relational structure and constraint satisfaction problems. In *In Proceedings of the 19th IEEE Annual Symposium on Logic in Computer Science (LICS'04), Turku, Finland*, 2004.

[10] A. Bulatov, P. Jeavons, and A. Krokhin. The complexity of constraint satisfaction: An algebraic approach (a survey paper). *In: Structural Theory of Automata, Semigroups and Universal Algebra (Montreal, 2003), NATO Science Series II: Mathematics, Physics, Chemistry*, 207:181–213, 2005.

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

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

[13] P. J. Cameron. Transitivity of permutation groups on unordered sets. *Math. Z.*, 148:127–139, 1976.

[14] D. A. Cohen, P. Jeavons, P. Jonsson, and M. Koubarakis. Building tractable disjunctive constraints. *J. ACM*, 47(5):826–853, 2000.

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

[16] V. Dalmau and J. Pearson. Closure functions and width 1 problems. *CP'99*, pages 159–173, 1999.

[17] 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.

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

[19] D. Geiger. Closed systems of functions and predicates. *Pacific Journal of Mathematics*, 27:95–100, 1968.

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

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

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

[23] H. Kautz, P. van Beek, and M. Vilain. Constraint propagation algorithms: A revised report. *Qualitative Reasoning about Physical Systems*, pages 373–381, 1990.

[24] 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.

[25] P.G.Jeavons and M.C.Cooper. Tractable constraints on ordered domains. *Artificial Intelligence*, 79(2):327–339, 1995.

[26] R.L.Graham, B. L. Rothschild, and J. H. Spencer. *Ramsey theory.* Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, Inc., 1990. Second edition.

[27] T. J. Schaefer. The complexity of satisfiability problems. In *Proceedings of STOC'78*, pages 216–226, 1978.

[28] A. Szendrei. *Clones in universal Algebra.* Seminaire de mathematiques superieures. Les Presses de L'Universite de Montreal, 1986.