Journal of Artificial Intelligence Research 58 (2017) 339-385 Submitted 07/16; published 02/17

A Model-Theoretic View on Qualitative Constraint Reasoning

Manuel Bodirsky MANUEL.BODIRSKY@TU-DRESDEN.DE
Institut für Algebra
TU Dresden
01062 Dresden, Germany

Peter Jonsson PETER.JONSSON@LIU.SE
Department of Computer Science
Linköping University
SE-581 83 Linköping, Sweden

Abstract

Qualitative reasoning formalisms are an active research topic in artificial intelligence. In this survey we present a model-theoretic perspective on qualitative constraint reasoning and explain some of the basic concepts and results in an accessible way. In particular, we discuss the significance of ω-categoricity for qualitative reasoning, of primitive positive interpretations for complexity analysis, and of Datalog as a unifying language for describing local consistency algorithms.

1. Introduction

This introductory section is divided into two parts: we present the background in the first part and outline the article in the second.

1.1 Background

Qualitative reasoning about physical systems has for a long time been an important and influential subarea of artificial intelligence. In particular, the intersection between qualitative reasoning and constraint reasoning has been a productive field. A large number of constraint-based formalisms for qualitative reasoning have been invented, most notably within temporal and spatial reasoning, and they have been investigated from many different angles such as modelling properties, computational complexity, and implementation issues. We will not attempt to survey this broad area but merely note that there are a number of introductory texts. General qualitative reasoning is covered by, for instance, the collection edited by Weld and de Kleer (1990) and the survey article by Dague (1995), while qualitative constraints is the topic of the textbook by Ligozat (2013). There are also two well-known surveys concerning temporal and spatial reasoning by Schwalb and Vila (1998) and Renz and Nebel (2007), respectively.

Research on qualitative constraint reasoning has concurrently been performed within the AI community and the theoretical computer science (TCS) community for many years. Unfortunately, collaboration and cross-fertilisation between the communities have been rare. This has led to a number of serious problems such as diverging terminology, rediscoveries of known results, and an ignorance of available methods and concepts. One notable example among many others is the *patchwork* property (Lutz & Milicic, 2007) which appears frequently in the literature on qualitative

constraint satisfaction problems (Amaneddine, Condotta, & Sioutis, 2013; Huang, 2012; Sioutis & Koubarakis, 2012): this property is essentially (under an appropriate translation between atomic networks and binary structures that will be explained later) the central notion of amalgamation in model theory. We thus find it highly desirable to have more communication between the two communities. We view this survey as a small step towards this goal.

The TCS community has to a large extent relied on methods from model theory and universal algebra when studying qualitative constraint problems. Briefly speaking, *model theory* is a branch of mathematics where mathematical structures are studied from a logic point of view: the textbooks by Hodges (1993) and Tent and Ziegler (2012) are excellent introductions to the topic. The aim of this survey is to present some of the powerful concepts from model theory that are directly applicable to qualitative constraint reasoning and to present them in an accessible way. In order to keep the presentation as concrete as possible, we will take a challenge posed by Renz (2012) as our point of departure. This allows us to introduce model-theoretic concepts and methods in a context that is familiar to the reader. Renz (2012, p. 517) writes the following:

> There are numerous examples of qualitative spatial or temporal calculi where the standard qualitative reasoning methods fail. There have been attempts to explain this behaviour, but it is largely unclear when and why this happens, how it can be avoided and what can be done about it. Due to the significance of being able to guarantee correct qualitative reasoning results, this is one of the major challenges in the field.

In order to address this challenge, it is necessary to contemplate upon two questions:

Q1. *What characterises qualitative CSPs?*

Q2. *What are the standard reasoning methods?*

A traditional AI answer would be that qualitative CSPs are those that can be defined by *partition schemes* (in the sense of Ligozat & Renz, 2004) while the reasoning methods are *consistency algorithms* such as arc consistency, path consistency, or more generally k-consistency. Both of these answers have been challenged lately: partition schemes have been criticised by, for instance, Westphal, Hué and Wölfl (2014) (see also Westphal, 2015), as being both too liberal and too restrictive, while the many different consistency notions call for a unified treatment (Bodirsky & Dalmau, 2013).

In this survey, we consider a quite narrow set of CSPs, namely those defined by ω-categorical constraint languages. The description "quite narrow set of CSPs" should not be taken too literally: it is a narrow set if viewed as a subset of all CSPs but it covers a substantial fraction of the CSPs that traditionally are referred to as qualitative. Let us give some examples. The point algebras for linear time, branching time, and partially ordered time can be formulated by using ω-categorical constraint languages (Bodirsky & Nešetřil, 2006). Hirsch (1996) has shown that these point algebras can be generalised to interval-based calculi in a systematic way that preserves ω-categoricity (and this is one of the many possible ways of proving ω-categoricity of Allen's Interval Algebra). Another way of generalising the point algebras is to study constraints that are first-order definable in them: one can show that if a constraint language Γ is ω-categorical, then every set of relations that is first-order definable in Γ is ω-categorical, too. In fact, there is an even stronger result: if a constraint language Γ is ω-categorical, then every set of relations that is *first-order interpretable* (see Section 3.4) in Γ is ω-categorical, too. Constructions of this kind are a natural source of interesting CSPs that are not

binary; see, for instance, the work by Bodirsky and Kára (2009) for concrete examples in temporal reasoning and the work by Kompatscher and Van Pham (2016) for examples based on partially ordered sets. If we turn our attention to spatial formalisms, then RCC-5 and RCC-8 can be formulated with ω-categorical constraint languages (Bodirsky & Chen, 2009; Bodirsky & Woelfl, 2011). Other examples (that were pointed out by Huang, 2012) are the n-dimensional block algebra (Balbiani, Condotta, & del Cerro, 1999) (where the *rectangle algebra* by Balbiani, Condotta, & del Cerro, 1998 is the 2-dimensional block algbera) and the *cardinal direction calculus* by Ligozat (1998)[1]. *Set constraints* are in certain ways connected to spatial formalisms but have also been used in description logics and program verification; Bodirsky and Hils (2012) have shown that set constraints can be formulated with ω-categorical constraint languages. There are also many examples from computer science outside artificial intelligence. A prominent example from bioinformatics is the rooted triple consistency problem that can be viewed as the CSP for a certain tree-like ω-categorical structure $(\mathbb{L}; C)$ (Bodirsky, Jonsson, & Pham, 2016). Interestingly, the relation C is ternary and therefore does not fit into the traditional framework of partition schemes of binary relations. Finally, we would like to mention that every problem in the complexity class MMSNP, a fragment of existential second-order logic that is studied in finite model theory, describes a CSP for an ω-categorical structure, or a finite union of CSPs for ω-categorical structures (Bodirsky & Dalmau, 2013). The class MMSNP contains a broad variety of combinatorial problems that sometimes show up in quite different contexts; e.g., it has been shown recently that every query in a natural class of computational problems in ontology-based data access is polynomial-time equivalent to a problem in MMSNP, and vice versa (Bienvenu, ten Cate, Lutz, & Wolter, 2014).

On the other hand, we consider a quite broad class of algorithms, namely those expressible in *Datalog*. This elegant framework for the formulation of consistency algorithms has been proposed by Feder and Vardi for constraint satisfaction problems over finite domains, and later by Bodirsky and Dalmau (2013) for qualitative infinite-domain CSPs. The class of algorithms contains, for instance, the usual k-consistency algorithms. An important advantage of this approach is that it is not restricted to partition schemes; in fact, it also works directly with constraints of larger arity.

Only considering a subclass of CSPs should obviously be viewed as a simplification of Renz's challenge, while considering a very broad range of algorithms is not necessarily a simplification. However, the connections between ω-categorical CSPs and Datalog are strong and, in many respects, quite well understood. Even under these additional assumptions the challenge by Renz remains difficult and is out of reach for current methods. However, when we restrict the class of CSPs under consideration further to the class of *finite-domain CSPs*, then the analog of Renz's challenge has recently been solved completely by Barto and Kozik (2014). This makes us slightly optimistic concerning the chances of solving the problem in the future.

1.2 Outline

We begin this survey with a fairly comprehensive section containing background material on CSPs (Section 2). In Sections 2.1 and 2.2, we link the terminology from temporal and spatial reasoning with the *homomorphism* terminology (that dominates the literature about the complexity of CSPs over finite domains) and the *sentence evaluation* terminology (which is natural from a logic point of view), respectively. These ways of defining CSPs are essentially only syntactical variants of each

1. The name 'cardinal connection calculus' has, unfortunately, been used also for another formalism by Goyal and Egenhofer (2001).

other but they nevertheless have distinct advantages, and this is the reason why we present both of them. Qualitative CSPs have traditionally been viewed as *network satisfaction problems* for *relation algebras* and we present this view together with a primer on relation algebras in Section 2.3. The network based view on CSPs is closely connected with sentence evaluation restricted to binary relations. This motivates Section 2.4 where we study the problem of converting CSPs with relations of arbitrary arity into CSPs with relations of arity two. Finally, we present two concrete relation algebras (Allen's Interval Algebra and RCC-5) in Section 2.5. The corresponding CSPs will be running examples throughout the survey. We use them for illustration purposes only; we have already seen that the results presented in this survey apply to a large collection of qualitative formalisms that have been discussed in the literature.

We continue in Section 3 by introducing ω-*categoricity*. There are strong connections between permutation groups and ω-categoricity and this explains the structure of this section: we begin by providing the basics of automorphisms and permutation groups in Section 3.1, and continue by introducing and exemplifying ω-categoricity in Section 3.2. In the next three sections (Sections 3.3–3.5), we present a number of concepts and techniques related to ω-categoricity; in particular, we introduce *(primitive positive) interpretations* which will prove to be a highly valuable tool.

After having introduced the necessary machinery, we can then start to discuss the question how to define qualitative CSPs. Ligozat and Renz (2004) have proposed a way of defining qualitative CSP by introducing *partition schemes* (see Section 4.1) that have been influential for qualitative constraint reasoning. This approach has recently been criticised (see, for instance, Westphal et al., 2014; Westphal, 2015) for being too restrictive in certain respects and too liberal in others. We review and discuss this criticism in Sections 4.2 and 4.3. This leads us (in Section 4.4) to an alternative way of formally defining (via ω-categoricity) what is meant by a qualitative CSP. We claim that this proposal overcomes some of the problems with the definition by Ligozat and Renz. However, we do not claim that this proposal is *the* right definition — this needs to be investigated more closely. The large number of qualitative CSPs that can be formulated with ω-categorical constraint languages is, however, a strong indication that ω-categoricity is a central ingredient when formally defining qualitative CSPs.

We give further evidence that ω-categoricity is an important property of qualitative CSPs in Section 5. There we show that every binary ω-categorical constraint language gives rise to an interesting finite proper relation algebra. This relation algebra contains in particular the relations of Γ and it is (naturally) closed under composition. Thus, there is no need to introduce the concept of *weak composition* (Renz & Ligozat, 2005) in order to use inference mechanisms such as the path consistency procedure. Weak composition is widespread within the literature on qualitative CSPs: it has been the dominant tool when dealing with "problematic" partition schemes that do not give rise to proper relation algebras.

The connection between Datalog and ω-categorical CSPs is the main theme of Section 6. We briefly introduce the syntax and semantics of Datalog in Section 6.1 together with an example based on the path consistency algorithm (Section 6.2). For CSPs over finite-domain or ω-categorical constraint languages, there exist so-called *canonical* Datalog programs that play an important role in many arguments about Datalog. Canonical programs will be introduced and explained in Section 6.3. In Section 6.4, we show that solvability by Datalog is preserved by primitive positive interpretability and discuss some implications of this. Finally, in Section 6.5 we present a general application of Datalog in qualitative reasoning, showing that the CSP for qualitative formalisms can be solved by Datalog (and consequently in polynomial time) when the input has bounded treewidth.

In Section 7, we return to Renz's challenge and present a criterion that implies that a CSP cannot be solved by Datalog. Informally speaking, the criterion is that the CSP can simulate satisfiability of linear equations over some finite Abelian group. For finite domain CSPs, this criterion is known to be necessary and sufficient due to a result by Barto and Kozik. This is no longer true for ω-categorical constraint languages: we will provide an example of an ω-categorical CSP which does not satisfy the criterion but still cannot be solved by Datalog due to a result by Bodirsky and Kára (2010). We conclude that it is not clear how to generalise the result of Barto and Kozik to qualitative infinite-domain CSPs. Despite the fact that it seems difficult to characterise the qualitative CSPs that can be solved by Datalog and even more difficult to fully resolve Renz's challenge, there are still plenty of reasonable research projects. In the last section of this survey, we describe some future research directions that we believe are feasible.

2. Constraint Satisfaction Problems

This section is divided into five subsections: Section 2.1-2.3 introduce different ways of formally describing CSPs, Section 2.4 presents methods for turning arbitrary CSPs into CSPs with binary signatures, and Section 2.5 introduces two CSPs that will be used as our primary examples in the rest of the survey.

Before we begin, we need to recall some terminology from logic. First-order formulas φ over the signature τ (or, in short, τ-formulas) are inductively defined using the logical symbols of universal and existential quantification, disjunction, conjunction, negation, equality, bracketing, variable symbols and the symbols from τ. The semantics of a first-order formula over some τ-structure is defined in the usual Tarskian style. A τ-formula without free variables is called a τ-sentence. We write $\Gamma \models \varphi$ iff the τ-structure Γ is a model for the τ-sentence φ, that is, satisfies φ; this notation is lifted to sets of sentences in the usual way.

One can use first-order formulas over the signature τ to define relations over a given τ-structure Γ: for a formula $\varphi(x_1, \ldots, x_k)$ where $x_1, \ldots, x_k$ are the free variables of φ the corresponding relation R is the set of all k-tuples $(t_1, \ldots, t_k) \in D_\Gamma^k$ such that $\varphi(t_1, \ldots, t_k)$ is true in Γ. In this case we say that R is *first-order definable* over Γ. Note that our definitions are always *parameter-free*, i.e. we do not allow the use of domain elements in them.

2.1 Constraint Satisfaction via Homomorphisms

We begin by presenting CSPs in terms of homomorphisms. A *relational signature* τ is a set of *relation symbols* R_i (also called *predicates*), each associated with an *arity* $k_i \in \mathbb{N}$. A *(relational) structure* Γ *over relational signature* τ (also called τ*-structure*) is a set D_Γ (the *domain*) together with a relation $R_i^\Gamma \subseteq D_\Gamma^{k_i}$ for each relation symbol R_i of arity k_i. If the reference to the structure Γ is clear, we may omit the superscript in R_i^Γ. For a τ-structure Γ and $R \in \tau$ it will also be convenient to say that $R(u_1, \ldots, u_k)$ *holds in* Γ iff $(u_1, \ldots, u_k) \in R$. We sometimes use the shortened notation $\overline{x}$ for a vector $x_1, \ldots, x_n$ of any length.

Let Γ and Δ be τ-structures. A *homomorphism* from Γ to Δ is a function f from D_Γ to D_Δ such that for each n-ary relation symbol R in τ and each n-tuple $\overline{a} = (a_1, \ldots, a_n)$, if $\overline{a} \in R^\Gamma$, then $(f(a_1), \ldots, f(a_n)) \in R^\Delta$. In this case we say that the map f *preserves* R.

Let Γ be a (possibly infinite) structure with a finite relational signature τ. Then the *constraint satisfaction problem (CSP)* for Γ is the following computational problem.

$\mathrm{CSP}(\Gamma)$
INSTANCE: A τ-structure Δ over a finite domain.
QUESTION: Is there a homomorphism from Δ to Γ?

In the homomorphism perspective on CSPs, the structure Γ is typically called the *template* of the constraint satisfaction problem $\mathrm{CSP}(\Gamma)$. The reader should be aware that several different names instead of the name template are used in the literature; the *constraint language* is probably the most common within AI.

A homomorphism from a given τ-structure Δ to Γ is called a *solution* of Δ for $\mathrm{CSP}(\Gamma)$. It is in general not clear how to represent solutions for $\mathrm{CSP}(\Gamma)$ on a computer; however, for the definition of the problem $\mathrm{CSP}(\Gamma)$ we do not need to represent solutions, since we only have to decide the *existence* of solutions. To represent an input structure Δ of $\mathrm{CSP}(\Gamma)$ we can fix any representation of the relation symbols in the signature τ, due to the assumption that τ is *finite*. Thus, $\mathrm{CSP}(\Gamma)$ is a well-defined computational problem for *any* infinite structure Γ with finite relational signature.

Example 1. (k-coloring problems). For $k \geq 1$, the k-colouring problem is the computational problem of deciding for a given finite graph G whether the vertices can be coloured by k colours such that adjacent vertices get different colours. It is a well-known fact that the k-colouring problem is NP-hard for $k \geq 3$ and in P for $k \leq 2$. For $k \geq 1$, let K_k denote the complete loop-free graph on k vertices. We view undirected graphs as τ-structures where τ contains a single binary relation symbol E which denotes a symmetric and anti-reflexive relation. Then the k-colorability problem can be modelled as $\mathrm{CSP}(K_k)$.

Example 2. (Digraph acyclicity). Consider the problem $\mathrm{CSP}(\mathbb{Z}; <)$. The binary relation $<$ denotes the strict linear order of the integers $\mathbb{Z}$. An instance G of this problem can be viewed as a directed graph (also called digraph), potentially with loops. It is easy to see that G homomorphically maps to $(\mathbb{Z}; <)$ if and only if there is no directed cycle in G (where loops are considered to be directed cycles, too). This can be decided in linear time in the size of the input, for example by performing a depth-first search on the digraph G.

Let Γ and Δ be two templates. We see that $\mathrm{CSP}(\Gamma)$ and $\mathrm{CSP}(\Delta)$ are the same computational problems if for every structure Θ, there is a homomorphism from Θ to Γ if and only if there is a homomorphism from Θ to Δ. If so, we simply say that Γ and Δ have the same CSP. One may, for instance, note that $\mathrm{CSP}(\mathbb{Z}; <)$ and $\mathrm{CSP}(\mathbb{R}; <)$ are the same computational problems and $(\mathbb{Z}; <)$ and $(\mathbb{R}; <)$ have the same CSP.

2.2 Constraint Satisfaction via Sentence Evaluation

We will now present the sentence evaluation view on CSPs. Let τ be a relational signature. A first-order τ-formula $\phi(x_1, \ldots, x_n)$ is called *primitive positive* if it is of the form

$$\exists x_{n+1}, \ldots, x_m(\psi_1 \wedge \cdots \wedge \psi_l)$$

where $\psi_1, \ldots, \psi_l$ are *atomic τ-formulas*, i.e., formulas of the form

1. $R(y_1, \ldots, y_k)$ with $R \in \tau$ and each $y_i \in \{x_1, \ldots, x_m\}$,
2. $y = y'$ for $y, y' \in \{x_1, \ldots, x_m\}$, or

3. $\bot$ (for *false*).

As usual, formulas without free variables are called *sentences*.[2] Let Γ be a (possibly infinite) structure with a finite relational signature τ. We can now define CSPs as follows.

$\mathrm{CSP}(\Gamma)$
INSTANCE: A primitive positive τ-sentence ϕ.
QUESTION: Is ϕ true in Γ?

It is easy to see that this definition is essentially the same as the previously presented definition based on homomorphisms and that the differences are a matter of formalization. A small difference between the homomorphism perspective and the sentence evaluation perspective follows from the fact that we *do* allow equality in primitive positive formulas. Adding equality to the constraint language does not affect the complexity of the CSP up to log-space reductions. However, there is research concerned with the complexity of CSPs at an even finer level than logspace-reducibility and, in these cases, equality is (naturally) not automatically allowed in the input to a constraint satisfaction problem.

The given primitive positive τ-sentence ϕ is often called an *instance* of $\mathrm{CSP}(\Gamma)$. The conjuncts of an instance ϕ are called the *constraints* of ϕ. A mapping from the variables of ϕ to D_Γ that is a satisfying assignment for the quantifier-free part of ϕ is also called a *solution* to ϕ.

Some authors omit the (existential) quantifier-prefix in instances ϕ of $\mathrm{CSP}(\Gamma)$ and the question is then whether ϕ is *satisfiable* over Γ or not. Clearly, this is just a rephrasing of the problem above but it explains the terminology of *satisfiable* and *unsatisfiable* (rather than *true* and *false*) instances of $\mathrm{CSP}(\Gamma)$.

It is well-known (Jeavons, 1998) that if Γ is a structure and a relation R has a primitive positive definition in Γ, then there is a polynomial-time reduction from $\mathrm{CSP}(\Gamma, R)$ to $\mathrm{CSP}(\Gamma)$. Here, (Γ, R) denotes the structure that we obtain from Γ by adding R as a new relation. The proof of this fact is almost trivial if one has the sentence evaluation view on CSPs. This may be something to keep in mind when we consider more complex methods for obtaining reductions between CSPs in Section 3.4.

Example 3 (Boolean satisfiability problems)**.** There are many Boolean satisfiability problems that can be cast as CSPs. Well-known examples are 3SAT (see Figure 1), and the restricted versions of 3SAT called 1-in-3-3SAT and NOT-ALL-EQUAL-3SAT (Garey & Johnson, 1978). These three problems are NP-complete. An interesting feature of the last two problems is that they remain NP-complete even when all clauses in the input only contain positive literals. With this additional restriction, the problems are called positive 1-in-3-3SAT and positive NOT-ALL-EQUAL-3SAT, and their definitions can be found in Figure 1. All of these problems can be formulated as $\mathrm{CSP}(\Gamma)$ for an appropriate 2-element structure Γ. Positive 1-in-3-3SAT can be formulated as $\mathrm{CSP}(\Gamma)$ for the template

$$\Gamma := (\{0,1\}; \mathrm{1IN3}) \quad \text{where } \mathrm{1IN3} = \{(0,0,1),(0,1,0),(1,0,0)\} ,$$

and Positive-Not-All-Equal-3SAT as $\mathrm{CSP}(\Gamma)$ for the template

$$\Gamma := (\{0,1\}, \mathrm{NAE}) \quad \text{where } \mathrm{NAE} = \{0,1\}^3 \setminus \{(0,0,0),(1,1,1)\} .$$

2. We do not need a symbol $\top$ for *true* since we can use the primitive positive formula $x = x$, for a new existentially quantified variable x, to express it.

3SAT
INSTANCE: A propositional formula in conjunctive normal form (CNF) with at most three literals per clause
QUESTION: Is there a Boolean assignment for the variables such that in each clause at least one literal is true?

Positive 1-in-3-3SAT
INSTANCE: A propositional 3SAT formula with only positive literals
QUESTION: Is there a Boolean assignment for the variables such that in each clause exactly one literal is true?

Positive Not-All-Equal-3SAT
INSTANCE: A propositional 3SAT formula with only positive literals
QUESTION: Is there a Boolean assignment for the variables such that in each clause neither all three literals are true nor all three are false?

Figure 1: Three Boolean satisfiability problems that can be formulated as $\mathrm{CSP}(\Gamma)$ for appropriate Γ

These problems can also be formulated as CSPs if we do *not* impose the restriction that all literals are positive; the idea is to use a different ternary relation for each of the eight ways how three distinct variables in a clause with three literals might be negated. The details are left to the reader. □

Example 4 (Disequality constraints)**.** Consider the problem $\mathrm{CSP}(\mathbb{N}; =, \neq)$. An instance of this problem can be viewed as an (existentially quantified) set of variables where some variables are linked by equality, some are linked by disequality[3], and some are not linked at all. Such an instance is false in $(\mathbb{N}; =, \neq)$ if and only if there is a path $x_1, \ldots, x_n$ from a variable x_1 to a variable x_n that uses only equality edges, i.e., '$x_i = x_{i+1}$' is a constraint in the instance for each $1 \leq i \leq n-1$, and additionally '$x_1 \neq x_n$' is a constraint in the instance. Clearly, it can be tested in linear time in the size of the input whether the instance contains such a path or not. □

2.3 Relation Algebras and Network Satisfaction

Qualitative CSPs have often been described using terminology from relation algebras. We give a short introduction to this topic and follow the presentation by Hirsch (1997). This approach is limited to formalisms with binary relations.

Definition 5. A *proper relation algebra* is a domain D together with a set $\mathcal{B}$ of binary relations over D such that

1. $\mathrm{Id} := \{(x, x) \mid x \in D\} \in \mathcal{B}$;

3. We deliberately use the word *disequality* instead of *inequality*, since we reserve the word *inequality* for the relation $\leq$.

2. If B_1 and B_2 are from $\mathcal{B}$, then $B_1 \vee B_2 := B_1 \cup B_2 \in \mathcal{B}$;
3. $1 := \bigcup_{R \in \mathcal{B}} R \in \mathcal{B}$;
4. $0 := \varnothing \in \mathcal{B}$;
5. If $B \in \mathcal{B}$, then $-B := 1 \setminus B \in \mathcal{B}$;
6. If $B \in \mathcal{B}$, then $B^\smile := \{(x, y) \mid (y, x) \in B\} \in \mathcal{B}$;
7. If B_1 and B_2 are from $\mathcal{B}$, then $B_1 \circ B_2 \in \mathcal{B}$; where

$$B_1 \circ B_2 := \{(x, z) \mid \exists y((x, y) \in B_1 \wedge (y, z) \in B_2)\} .$$

We want to point out that in this standard definition of proper relation algebras it is *not* required that 1 denotes D^2. However, in most examples, 1 indeed denotes D^2. Furthermore, we always have implicit closure under intersection since the intersection of two relations B_1, B_2 equals $-(-B_1 \vee -B_2)$. The inclusion-wise minimal non-empty elements of $\mathcal{B}$ are called the *basic relations* of the relation algebra.

Example 6 (The Point Algebra)**.** Let $D = \mathbb{Q}$ be the set of rational numbers, and consider

$$\mathcal{B} = \{<, >, =, \leq, \geq, \neq, \varnothing, \mathbb{Q}^2\} .$$

These relations form a proper relation algebra (with the three atoms $<, >, =$ and where 1 denotes $\mathbb{Q}^2$). This is one of the most fundamental relation algebras and it is known under the name *point algebra*. □

When $\mathcal{B}$ is finite, every relation in $\mathcal{B}$ can be written as a finite union of basic relations. We sometimes abuse notation and write $R = \{B_1, \ldots, B_k\}$ when $B_1, \ldots, B_k$ are basic relations, $R \in \mathcal{B}$, and $R = B_1 \cup \cdots \cup B_k$. Note that composition of basic relations determines the composition of all relations in the relation algebra, since

$$R_1 \circ R_2 = \bigcup_{B_1 \in R_1, B_2 \in R_2 \text{ basic}} B_1 \circ B_2 .$$

An *abstract relation algebra* is an algebra with the signature $\{\vee, -, 0, 1, \circ, ^\smile, \mathrm{Id}\}$ that satisfies the laws that we would expect for the respective operators in a proper relation algebra. The formal machinery may vary slightly in its details (Hirsch, 1997; Düntsch, 2005; Ladkin & Maddux, 1994) but the essence is captured by the following definition.

Definition 7. By an abstract relation algebra $\mathbf{A}$, we mean an algebra with domain A and signature $\{\vee, -, 0, 1, \circ, ^\smile, \mathrm{Id}\}$ such that

- the structure $(A; \vee, \wedge, -, 0, 1)$ is a Boolean algebra where $\wedge$ is defined by $(x, y) \mapsto -(-x \vee -y)$ from $-$ and $\vee$;
- $\circ$ is an associative binary operation on A;
- $(a^\smile)^\smile = a$ for all $a \in A$;

$\circ$	$=$	$<$	$>$
$=$	$=$	$<$	$>$
$<$	$<$	$<$	1
$>$	$>$	1	$>$

Figure 2: The composition table for the basic relations in the point algebra.

- $a \circ (b \vee c) = a \circ b \vee a \circ c$;
- $(a \vee b)^\smile = a^\smile \vee b^\smile$;
- $(-a)^\smile = -(a^\smile)$;
- $(a \circ b)^\smile = b^\smile \circ a^\smile$;
- $(a \circ b) \wedge c^\smile = 0 \;\Leftrightarrow\; (b \circ c) \wedge a^\smile = 0$.

The minimal elements of A with respect to the order $\leq$ defined by $x \leq y \Leftrightarrow x \wedge y = x$ are called the *atoms* of $\mathbf{A}$.

A *representation* (D, i) of $\mathbf{A}$ consists of a set D and a mapping i from the domain A of $\mathbf{A}$ to the set of binary relations over D such that the image of i induces a proper relation algebra $\mathbf{B}$, and i is an isomorphism with respect to the functions (and constants) $\{\vee, -, 0, 1, \circ, ^\smile, \mathrm{Id}\}$. In this case, we also say that $\mathbf{A}$ is the *abstract relation algebra of* $\mathbf{B}$.

Example 8. The (abstract) point algebra is a relation algebra with 8 elements and 3 atoms, $=$, $<$, and $>$, and can be described as follows. The composition operator of the basic relations of the point algebra is shown in the table of Figure 2. By the observation we just made, this table determines the full composition table. The inverse of $<$ is $>$, and Id denotes $=$ which is its own inverse. This fully determines the relation algebra. We can obtain a representation of the abstract point algebra from the point algebra with domain $\mathbb{Q}$ presented in Example 6 in the obvious way. □

The central computational problems that have been studied for relation algebras are *network satisfaction problems* (Ladkin & Maddux, 1994; Düntsch, 2005; Hirsch, 1997).

Definition 9. Let $\mathbf{A}$ be a finite relation algebra with domain A. An *($\mathbf{A}$-) network* $N = (V; f)$ consists of a finite set of nodes V and a partial function $f\colon V^2 \to A$.

The network is called *atomic* if f is defined for all pairs of nodes and the image of f only contains atoms of $\mathbf{A}$. For convenience and according to common practice, instead of $f(u, v) = (b_1 \vee \cdots \vee b_n)$ we say that N *contains the constraint* $u\{b_1, \ldots, b_n\}v$.

Two types of network satisfaction problems have been studied for $\mathbf{A}$-networks. We start with the network satisfaction problem for a given fixed representation.

Definition 10. Let (D, i) be a representation of a finite relation algebra $\mathbf{A}$. Then the *network satisfaction problem for* (D, i) is the computational problem of deciding whether a given $\mathbf{A}$-network N is *satisfiable with respect to* (D, i), i.e., whether there exists a mapping $s\colon V \to D$ such that $(s(u), s(v)) \in i(f(u, v))$ for all $u, v \in V$ where $f(u, v)$ is defined.

Note that for interpretations (D, i) where $i(1) = D^2$ we could have required that the function f in the definition of networks is total, since setting $f(u, v) = 1$ would then be equivalent to leaving f undefined. However, since in general $i(1)$ need not be equal to D^2 (following the standard definition), allowing f to be partial might have an effect on the network satisfaction problem for (D, i).

The second problem is the *(general) network satisfaction problem for* $\mathbf{A}$.

Definition 11. Let $\mathbf{A}$ be a finite relation algebra. Then the *network satisfaction problem for* $\mathbf{A}$ is the computational problem to decide whether for a given $\mathbf{A}$-network N there *exists* a representation (D, i) of $\mathbf{A}$ such that N is satisfiable with respect to (D, i).

It is not surprising that every network satisfaction problem for a fixed representation is closely related to a corresponding constraint satisfaction problem; this correspondence will be described in the following. It is maybe less obvious that the same also applies to the *general* network satisfaction problem: every finite relation algebra $\mathbf{A}$ that has a representation also has a representation (D, i) such that the general network satisfaction problem for $\mathbf{A}$ and the network satisfaction problem for (D, i) are one and the same problem (Proposition 13).

To present the link between network satisfaction problems and CSPs as defined earlier we need the following notation. Let $\tau_{\mathbf{A}}$ be a signature consisting of a binary relation symbol R_a for each element $a \in A$. When (D, i) is a representation of $\tau_{\mathbf{A}}$, then this gives rise to a $\tau_{\mathbf{A}}$-structure $\Gamma_{D,i}$ in a natural way: the domain of the structure is D and the relation symbol R_a is interpreted as $i(a)$. We can associate to each $\mathbf{A}$-network $N = (V; f)$ a primitive positive $\tau_{\mathbf{A}}$-sentence ϕ_N, in the following straightforward way: the variables of ϕ_N are V and ϕ_N contains the conjunct $R_a(u, v)$ iff $f(u, v) = a$. Conversely, we can associate to each primitive positive $\tau_{\mathbf{A}}$-sentence ϕ with variables V a network N_ϕ as follows. Assume that the domain of N_ϕ is V. Let $u, v \in V$ and let $a_1, \ldots, a_k$ be the elements a of A such that ϕ contains the conjunct $R_a(u, v)$. Then define $f(u, v) = a$ for $a = (a_1 \wedge a_2 \wedge \cdots \wedge a_k)$; if $k = 0$, then f is undefined for u, v.

The following link between the network satisfaction problem for a fixed representation (D, i) of $\mathbf{A}$ and the constraint satisfaction problem for $\Gamma_{D,i}$ is straightforward to prove from the definitions.

Proposition 12. *Let* $\mathbf{A}$ *be a finite relation algebra with representation* (D, i)*. Then an* $\mathbf{A}$*-network* N *is satisfiable with respect to* (D, i) *if and only if* $\Gamma_{D,i} \models \phi_N$*. Conversely,* $\Gamma_{D,i}$ *satisfies a primitive positive* $\tau_{\mathbf{A}}$*-sentence* ϕ *if and only if* N_ϕ *is satisfiable with respect to* (D, i)*.*

Proposition 12 shows that network satisfaction problems for fixed representations essentially *are* constraint satisfaction problems and that the differences are only a matter of formalization. To also relate the *general* network satisfaction problem for a finite relation algebra $\mathbf{A}$ to a constraint satisfaction problem, we use the following.

Proposition 13 (Bodirsky, 2012, Proposition 1.3.16)**.** *Every finite relation algebra* $\mathbf{A}$ *that has a representation also has a (countable) representation* (D, i) *whose network satisfaction problem is the same problem as the general network satisfaction problem for* $\mathbf{A}$*.*

In combination with Proposition 12, this implies that also every general network satisfiability problem is essentially the same problem as a CSP for a countably infinite template.

2.4 Binary Signatures

Most constraint formalisms that have been studied in the qualitative reasoning literature only involve *binary* constraints. Typically, this is no major limitation since there is a well-known general reduction from a constraint language Γ with arbitrary arities to a binary constraint language Δ such that the corresponding CSPs are closely related; in particular, they are polynomial-time equivalent. This reduction is known under the name *dual encoding* in the literature (Dechter, 2003; Cohen, Cooper, Jeavons, & Zivny, 2015).

We want to stress that this transformation works for arbitrary finite constraint languages over a finite *or infinite* domain D. For simplicity, we assume that the constraint language consists of a single k-ary relation R; the construction for general constraint languages with finitely many relations is analogous but notationally more cumbersome.

The domain of the constraint language Δ is D^k. The binary signature of Δ includes the unary relation R, and for all $i, j \in \{1, \dots, k\}$ a binary relation symbol $E_{i,j}$ which denotes the relation

$$\left\{((x_1, \dots, x_k), (y_1, \dots, y_k)) \mid x_1, \dots, x_k, y_1, \dots, y_k \in D \text{ and } x_i = y_j\right\}.$$

Proposition 14. *$\mathrm{CSP}(D; R)$ and $\mathrm{CSP}(\Delta)$ are polynomial-time equivalent.*

Proof. We first describe how to reduce $\mathrm{CSP}(D; R)$ to $\mathrm{CSP}(\Delta)$, and thereafter the reduction in the other direction. Given a primitive positive sentence

$$\phi = \exists \bar{x} \left(R(x_1^1, \dots, x_k^1) \wedge \cdots \wedge R(x_1^m, \dots, x_k^m)\right)$$

we construct the primitive positive formula ψ with variables $y_1, \dots, y_m$ that has the following conjuncts:

- $E_{i,j}(y_p, y_q)$ for all $i, j \leq k$ and $p, q \leq m$ with $x_i^p = x_j^q$.
- $R(y_l)$ for all $k \in \{1, \dots, m\}$.

Claim 1.1. When ϕ is satisfiable in $(D; R)$ then ψ is satisfiable in Δ.
Let $s \colon \{x_1^1, \dots, x_k^m\} \to D$ be an assignment that satisfies the quantifier-free part of ϕ. Then $(s(x_1^i), \dots, s(x_k^i)) \in R$ for each $i \in \{1, \dots, m\}$ and the map that sends y_i to $(s(x_1^i), \dots, s(x_k^i))$ satisfies all conjuncts of ψ.

Claim 1.2. When ψ is satisfiable in Δ then ϕ is satisfiable in $(D; R)$.
Let $t \colon \{y_1, \dots, y_m\} \to R$ be an assignment that satisfies the quantifier-free part of ψ. Then the map s that assigns x_j^i to the j-th entry of $t(y_i)$ is well-defined because t satisfies the conjuncts of ψ. Then $(s(x_1^i), \dots, s(x_k^i)) = t(y_i) \in R$ for all $i \in \{1, \dots, m\}$, and thus ϕ is satisfied, too.

We now describe the reduction from $\mathrm{CSP}(\Delta)$ to $\mathrm{CSP}(D; R)$. Given a primitive positive sentence ψ with m variables $y_1, \dots, y_m$. Then we construct a primitive positive sentence ϕ with the set of variables $V := \{x_1^1, \dots, x_k^1, \dots, x_1^m, \dots, x_k^m\}$ as follows. For each conjunct $E_{i,j}(y_p, y_q)$ of ψ we add the conjunct $x_i^p = x_j^q$. For each conjunct $R(y_l)$ we add the conjunct $R(x_1^i, \dots, x_k^i)$.

Claim 2.1. When ψ is satisfiable in Δ then ψ is satisfiable in $(D; R)$.
Let $t \colon \{y_1, \dots, y_m\} \to R$ be an assignment that satisfies the quantifier-free part of ψ. Let s be the map that assigns x_j^i to the j-th entry of $t(y_i)$. If $x_i^p = x_j^q$ is a conjunct of ϕ, then ψ must contain the conjunct $E_{i,j}(y_p, y_q)$, and since t satisfies this conjunct we must have $s(x_i^p) = s(x_j^q)$. Moreover, t satisfies conjuncts of the form $R(x_1^i, \dots, x_k^i)$ because $(s(x_1^i), \dots, s(x_k^i)) = t(y_i) \in R$.

Claim 2.2. When ϕ is satisfiable in $(D; R)$ then ψ is satisfiable in Δ.
Let $s\colon V \to D$ be an assignment that satisfies the quantifier-free part of ϕ. The map t that assigns y_i to $(s(x_1^i), \ldots, s(x_k^i))$ satisfies the constraints of the form $R(y_i)$ because s satisfies the conjuncts of the form $R(x_1^i, \ldots, x_k^i)$. Moreover, for each conjunct $E_{i,j}(y_p, y_q)$ of ψ the sentence ϕ has a conjunct $x_i^p = x_j^q$. Since s satisfies this conjunct, t satisfies $E_{i,j}(y_p, y_q)$. □

The dual encoding preserves not only the complexity of the CSP but many other important properties of $(D; R)$; we will come back to this point in Sections 3.4 and 6.4.

Other reductions to binary languages can be found in the literature and a well-known alternative to the dual encoding is the *hidden variable encoding* by Rossi, Petrie, and Dhar (1990) A quite different reduction has been given by Feder and Vardi (1999); their reduction even produces a constraint language with a *single* binary relation. However, this approach has only been described for finite domains.

2.5 Examples

We will now introduce and discuss two CSPs that have frequently been used in AI.

2.5.1 ALLEN'S INTERVAL ALGEBRA

Allen's Interval Algebra (Allen, 1983) is a formalism that is both well-known and well-studied in artificial intelligence. It was introduced to reason about intervals and the qualitative relationships between intervals. Formally, Allen's Interval Algebra is an abstract relation algebra (see Section 2.3); however, it is most natural to introduce it via one of its natural representations, namely the representation on the set $\mathbb{I}$ of all finite closed intervals $[a, b]$ of rational numbers, where $a, b \in \mathbb{Q}$ and $a < b$. If $I = [a, b] \in \mathbb{I}$, then we write I^- for a and I^+ for b.

Basic relation		Example	Endpoints
I precedes J	p	III JJJ	$I^+ < J^-$
J preceded by I	$\mathsf{p}^\smile$		
I meets J	m	IIII JJJJ	$I^+ = J^-$
J is met by I	$\mathsf{m}^\smile$		
I overlaps J	o	IIII JJJJ	$I^- < J^- < I^+$, $I^+ < J^+$
J overlapped by I	$\mathsf{o}^\smile$		
I during J	d	II JJJJJJ	$I^- > J^-$, $I^+ < J^+$
J includes I	$\mathsf{d}^\smile$		
I starts J	s	III JJJJJJ	$I^- = J^-$, $I^+ < J^+$
J started by I	$\mathsf{s}^\smile$		
I finishes J	f	III JJJJJJ	$I^+ = J^+$, $I^- > J^-$
J finished by I	$\mathsf{f}^\smile$		
I equals J	$\equiv$	IIII JJJJ	$I^- = J^-$, $I^+ = J^+$

Figure 3: The definitions of the basic relations of Allen's Interval Algebra.

$\circ$	PP	$\text{PP}^\smile$	DR	PO	EQ
PP	{PP}	*	{DR}	{PP, DR, PO}	{PP}
$\text{PP}^\smile$	{PP, $\text{PP}^\smile$, PO, EQ}	{$\text{PP}^\smile$}	{$\text{PP}^\smile$, DR, PO}	{$\text{PP}^\smile$, PO}	{$\text{PP}^\smile$}
DR	{PP, DR, PO}	{DR}	*	{PP, DR, PO}	{DR}
PO	{PP, PO}	{$\text{PP}^\smile$, DR, PO}	{$\text{PP}^\smile$, DR, PO}	*	{PO}
EQ	{PP}	{$\text{PP}^\smile$}	{DR}	{PO}	{EQ}

Figure 4: The composition table of RCC-5. An asterisk * represents the set $\{\text{PP}, \text{PP}^\smile, \text{DR}, \text{PO}, \text{EQ}\}$ and $\{B_1, \ldots, B_k\}$ represents the relation $B_1 \cup \cdots \cup B_k$.

$$
\begin{array}{lll}
X\{\text{PP}\}Y & \text{iff} & X \subset Y \\
X\{\text{PP}^\smile\}Y & \text{iff} & X \supset Y \\
X\{\text{DR}\}Y & \text{iff} & X \cap Y = \varnothing \\
X\{\text{PO}\}Y & \text{iff} & \exists a, b, c : a \in X, a \notin Y, b \in X, b \in Y, c \notin X, c \in Y \\
X\{\text{EQ}\}Y & \text{iff} & X = Y
\end{array}
$$

Figure 5: Basic relations of RCC-5_{disk}.

The basic relations in this representation of Allen's Interval Algebra are the 13 relations defined in Figure 3. The network consistency problem for this proper relation algebra equals the general network satisfaction problem for Allen's Interval Algebra (Ladkin & Maddux, 1994), and the computational complexity of CSP(A) for all 2^{8192} subsets A of Allen's relations is known (Krokhin, Jeavons, & Jonsson, 2003).

2.5.2 RCC-5

The RCC formalisms (Randell, Cui, & Cohn, 1992) are designed for reasoning about spatial regions and they are the basis for a large part of the work in *qualitative spatial reasoning* (QSR). There are several variants such as RCC-23, RCC-8, and RCC-5. We concentrate on the simplest formalism, RCC-5, in this survey. By definition, RCC-5 is an abstract relation algebra, given by the table shown in Figure 4. The complexity of the network satisfaction problem for all subsets of relations are known (Jonsson & Drakengren, 1997). In RCC-5, PP stands for *proper part*, PO stands for *partial overlap*, DR stands for *disconnected regions*, and EQ stands for *equality*. It is in many ways more intuitive to introduce RCC-5 via one of its representations than via its abstract relation algebra. Below, we will discuss some representations of RCC-5 and the corresponding CSPs.

Let us begin with a domain consisting of the nonempty open disks in $\mathbb{R}^2$. The interpretation of the base elements of RCC-5 as binary relations over this domain is given in Figure 5. Düntsch et al. (1999, bottom of page 235) have proved that this is a representation of RCC-5 and we denote it by RCC-5_{disk}.

Next, we present another representation of RCC-5. This representation is based on the standard representation of the spatial calculus RCC-8 (Renz & Nebel, 2001, Sec. 3.1) but where one is not able to distinguish regions from their topological closure: the disconnectedness relations DC and EC are replaced by DR and the tangential and non-tangential proper part relations TPP and NTPP are replaced by PP (and the relation $\text{PP}^\smile$ is introduced analogously). Here, the domain consists of the

$$
\begin{array}{lll}
X\{\mathtt{PP}\}Y & \text{iff} & X \subset \mathrm{int}(Y) \\
X\{\mathtt{PP}^\smile\}Y & \text{iff} & \mathrm{int}(X) \supset Y \\
X\{\mathtt{DR}\}Y & \text{iff} & \mathrm{int}(X) \cap \mathrm{int}(Y) = \varnothing \\
X\{\mathtt{PO}\}Y & \text{iff} & \mathrm{int}(X) \cap \mathrm{int}(Y) \neq \varnothing, X \not\subseteq Y, Y \not\subseteq X \\
X\{\mathtt{EQ}\}Y & \text{iff} & X = Y
\end{array}
$$

Figure 6: Basic relations of RCC-5_{reg} where $\mathrm{int}(\cdot)$ denotes the interior operator.

nonempty regular closed subsets of some regular and connected topological space. A subset of a topological space is called *regular closed* if it is equal to the closure of its interior. Note that the sets are *not* required to be connected and this is an important difference between RCC-5_{reg} and RCC-5_{disk} where each domain element is a connected set by definition. Now, the interpretation of the basic relations of RCC-5 is given by Figure 6. The resulting proper relation algebra is known to be a representation of RCC-5, (Renz & Nebel, 2007, Sec. 2). We henceforth call this algebra RCC-5_{reg}. Note that the interior of an open disk in $\mathbb{R}^2$ is the open disk itself. Hence, the RCC-5 relations for RCC-5_{disk} can equivalently be defined as in Figure 6 (albeit in a slightly more cumbersome way).

We will now compare RCC-5_{disk} and RCC-5_{reg}. Let $x_1, \ldots, x_6$ be variables and constrain them as follows:

$$x_1\{\mathtt{PO}\}x_2, x_2\{\mathtt{PO}\}x_3, x_3\{\mathtt{PO}\}x_4, x_4\{\mathtt{PO}\}x_1,$$

$$x_1\{\mathtt{DR}\}x_3, x_2\{\mathtt{DR}\}x_4$$

In other words, the regions $x_1, \ldots, x_4$ form a "cycle" in the plane. Now, suppose that additionally the following holds.

$$x_5\{\mathtt{PO}\}x_i, x_6\{\mathtt{PO}\}x_i \text{ for } 1 \leq i \leq 4$$

It is not hard to see that in RCC-5_{disk}, it is impossible that $x_5\{\mathtt{DR}\}x_6$ holds, while it is possible in RCC-5_{reg}. It follows that the CSPs for RCC-5_{reg} and RCC-5_{disk} are different. Note here that RCC-5_{reg} and RCC-5_{disk} have exactly the same composition table so the difference cannot be inferred from the table alone.

We have seen that different representations of a relation algebra might have different CSPs. Let us illustrate that conversely there are binary structures with the same CSP where one structure forms a proper relation algebra but the other structure does not. The example is based on ideas by Li et al. (2015, Sec. 2.2). We now consider arbitrary non-empty subsets of an infinite set such as $\mathbb{N}$. We define the relations $\mathtt{PP}, \mathtt{PP}^\smile, \mathtt{DR}, \mathtt{PO}, \mathtt{EQ}$ as in Figure 5 and denote the resulting structure by RCC-5_{set}. We verify that the relations of the structure RCC-5_{set} do not form a proper relation algebra by showing that the composition $\mathtt{DR} \circ \mathtt{DR}$ does not belong to a finite union of relations from $\mathtt{PP}, \mathtt{PP}^\smile, \mathtt{DR}, \mathtt{PO}, \mathtt{EQ}$. Let $a = \{1, 2\}$, $b = \{3\}$, and $c = \{2, 4\}$ and note that $a\{\mathtt{PO}\}c$, $a\{\mathtt{DR}\}b$, and $b\{\mathtt{DR}\}c$. By the definition of the composition operator $\circ$, this implies that $\mathtt{PO} \cap \mathtt{DR} \circ \mathtt{DR}$ is non-empty. Now let $d = \mathbb{N} \setminus \{1\}$, $e = \mathbb{N} \setminus \{2\}$, and note that $d\{\mathtt{PO}\}e$ and $d \cup e = \mathbb{N}$. Thus, there is no non-empty $f \subseteq \mathbb{N}$ such that $d\{\mathtt{DR}\}f$ and $f\{\mathtt{DR}\}e$ hold simultaneously. We conclude that $\mathtt{PO} \cap (\mathtt{DR} \circ \mathtt{DR}) \neq \varnothing$ and $\mathtt{PO} \not\subseteq (\mathtt{DR} \circ \mathtt{DR})$ so $(\mathtt{DR} \circ \mathtt{DR})$ is not an RCC-5 relation. Despite this, we have the following result.

Proposition 15. *Let N be an RCC-5 network. Then the following are equivalent.*

1. *N is satisfiable in RCC-5_{reg};*

2. *N is satisfiable in some representation of RCC-5;*

3. *the structure RCC-5$_{\mathrm{set}}$ satisfies ϕ_N (as defined before Proposition 12).*

Proof. We write $V = \{v_1, \ldots, v_n\}$ for the nodes of N. The implication from 1 to 2 is immediate since RCC-5$_{\mathrm{reg}}$ is a representation of RCC-5 as mentioned above.

For the implication from 2 to 3, assume that N has a solution s in some representation (D, i) of RCC-5. We will show that s is a satisfying assignment for ϕ_N in the structure RCC-5$_{\mathrm{set}}$. The following proof is based on an idea by Drakengren and Jonsson (1998). Construct a satisfiable atomic network N' as follows.

Step 1. Replace each constraint $x\{b_1, \ldots, b_k\}y$ in N by $x\{b\}y$ where $b \in \{b_1, \ldots, b_k\}$ such that $(s(x), s(y)) \in i(b)$.

Step 2. Remove every constraint of the type $x\{\mathtt{EQ}\}y$: this can be done by collapsing the variables x and y (we leave the obvious details of this step to the reader).

Step 3. Replace every constraint of the type $x\{\mathtt{PO}\}y$ with the constraints $z_1\{\mathtt{DR}\}z_2$, $z_2\{\mathtt{DR}\}z_3$, $z_3\{\mathtt{DR}\}z_1$, $z_1\{\mathtt{PP}\}x$, $z_1\{\mathtt{DR}\}y$, $z_2\{\mathtt{PP}\}x$, $z_2\{\mathtt{PP}\}y$, and $z_3\{\mathtt{PP}\}y$, $z_3\{\mathtt{DR}\}x$, where z_1, z_2, z_3 are fresh variables. The fact that (D, i) is an interpretation of RCC-5 implies that the relation $x\{\mathtt{PO}\}y$ is enforced by the constraints above. This can be verified by, for instance, using the composition table for RCC-5.

Let N' denote the resulting network and V' its nodes. We say that two variables u, v in N' are PP-*connected* if there exists a sequence of variables $w_1, \ldots, w_p$ such that $w_1 = u$, $w_p = v$, and $w_i\{\mathtt{PP}\}w_{i+1}$ for all $1 \leq i < p$. Note that if u and v are PP-connected, then in any solution s' of N' we have that $s'(u)\{\mathtt{PP}\}s'(v)$; this follows from the fact that (D, i) is an interpretation of RCC-5 and $\mathtt{PP} \circ \mathtt{PP} = \mathtt{PP}$. Now, define an assignment $s' \colon V' \to 2^{\mathbb{N}} \setminus \{\varnothing\}$ as follows:

$$s'(v_i) := \{i\} \cup \{k \mid v_k \in V \text{ is PP-connected to } v_i\}.$$

The map s' shows that the structure RCC-5$_{\mathrm{set}}$ satisfies $\phi_{N'}$. The main observation is the following: if $x\{\mathtt{DR}\}y$ is a constraint in N', then there is no variable that is PP-connected to both x and y, and consequently $s'(x) \cap s'(y) = \varnothing$. Then the restriction of s' to V (with the obvious treatment of variables that have been collapsed) shows that ϕ_N holds in RCC-5$_{\mathrm{set}}$, too.

For the implication from 3 to 1, let $t_1, t_2, \ldots$ be an enumeration of all closed disks of radius $1/4$ around integer points in $\mathbb{R}^2$, and let T be the set that contains all finite unions of such disks. Note that all elements of T are regular closed in the space $\mathbb{R}^2$ equipped with the usual topology, and that this topological space is both regular and connected. Assume that $s \colon V \to 2^{\mathbb{N}} \setminus \{\varnothing\}$ is a map that satisfies all conjuncts of ϕ_N in RCC-5$_{\mathrm{set}}$. Define $s' \colon V \to T$ by $s'(v) := \bigcup_{i \in s(v)} t_i$. It is not difficult to verify that s' is a solution to N with respect to RCC-5$_{\mathrm{reg}}$. □

We summarise: if an RCC-5 network N is satisfiable in some representation of RCC-5, then it is satisfiable in RCC-5$_{\mathrm{reg}}$, too, and RCC-5$_{\mathrm{reg}}$ resembles the general network satisfaction problem in this respect. On the other hand, the general network satisfaction problem for RCC-5 is different from the network satisfaction problem for RCC-5$_{\mathrm{disk}}$. Finally, the structure RCC-5$_{\mathrm{set}}$ has the same CSP as RCC-5$_{\mathrm{reg}}$, but RCC-5$_{\mathrm{set}}$ is not a representation of RCC-5.

3. CSPs and ω-Categoricity

The model-theoretic concept of ω-categoricity plays a fundamental role in this survey and the official definition will be given in Section 3.2. There is an equivalent characterisation of ω-categoricity in terms of permutation groups, due to Engeler, Svenonius, and Ryll-Nardzewski, which is easier to work with and which might be viewed as a finiteness condition. In this survey, we will exclusively work with this permutation group characterisation of ω-categoricity. This explains the subdivision of this section: we begin in Section 3.1 by recalling basic concepts concerning permutation groups and we continue in Section 3.2 by introducing ω-categoricity. The remaining three sections are used for introducing some key concepts and methods that use ω-categoricity in different ways. For further reading about ω-categoricity, there is an excellent body of literature: examples include the book by Cameron (1990), the survey by Macpherson (2011), and the collection edited by Kaye and Macpherson (1994). Moreover, classical text-books on model theory, such as those by Hodges (1993) and Marker (2002), always treat ω-categority, and use ω-categorical structures as a rich source of examples.

3.1 Permutation Groups

In the following, let G be a set of permutations of a set X. We say that G is a *permutation group* if the identity permutation is in G and for arbitrary $g, f \in G$, the functions $x \mapsto g(f(x))$ and $x \mapsto g^{-1}(x)$ are also in G. In other words, G is closed under function composition and inversion. The set of all permutations of a set X is a permutation group that is known as the *symmetric group* $\mathrm{Sym}(X)$. Let Γ denote a τ-structure. An *automorphism* of Γ is a bijective homomorphism from Γ to Γ whose inverse is also a homomorphism. The proof of the following fact is straightforward.

Proposition 16. *Let Γ be a structure with domain D_Γ, and suppose that R is first-order definable over Γ. Then every automorphism of Γ is also an automorphism of $(D_\Gamma; R)$.*

The automorphisms of Γ form a permutation group $\mathrm{Aut}(\Gamma)$ on the set D_Γ.

Example 17. Consider the structure $(\mathbb{Z}; <)$. It is easy to verify that the automorphisms of $(\mathbb{Z}; <)$ are precisely the functions of the form $x \mapsto x + c$ where $c \in \mathbb{Z}$.

For $n \geq 1$, the *orbit* of $(t_1, \ldots, t_n) \in X^n$ under G is the set $\{(\alpha(t_1), \ldots, \alpha(t_n)) \mid \alpha \in G\}$. Clearly, the orbits of n-tuples under G partition the set X^n, that is, every $(t_1, \ldots, t_n) \in X^n$ lies in precisely one orbit under G.

3.2 Countably Categorical Structures

A (first-order) *theory* is a set T of first-order sentences. If the first-order sentences are over the signature τ, we say that T is a *τ-theory*. The *first-order theory* of a τ-structure Δ (denoted by $\mathrm{Th}(\Delta)$) is the set of τ-sentences ϕ such that $\Delta \models \phi$. A *model* of a τ-theory T is a τ-structure Δ such that Δ satisfies all sentences in T. Theories that have a model are called *satisfiable*. We can now formally define the central concept of this section: a satisfiable first-order theory T is *ω-categorical* if all countable models of T are isomorphic, and a structure is ω-categorical if its first-order theory is ω-categorical. Note that the first-order theory of a finite structure does not have infinite models so finite structures are always ω-categorical. One of the first infinite structures that were found to be ω-categorical (by Cantor, 1884) is the linear order of the rational numbers $(\mathbb{Q}; <)$.

There are many equivalent characterizations of ω-categoricity and the most important one is in terms of the automorphism group.

Definition 18. A permutation group G over a countably infinite set X is *oligomorphic* if G has only finitely many orbits of n-tuples for each $n \geq 1$.

An accessible proof of the following theorem can be found in Hodges' (1993) book.

Theorem 19 (Engeler, Ryll-Nardzewski, Svenonius)**.** *For a countably infinite structure Γ with countable signature, the following are equivalent:*

1. *Γ is ω-categorical;*
2. *the automorphism group $\mathrm{Aut}(\Gamma)$ of Γ is oligomorphic;*
3. *for each $n \geq 1$, there are finitely many inequivalent formulas with free variables $x_1, \ldots, x_n$ over Γ;*
4. *for all $n \geq 1$, every set of n-tuples that is preserved by all automorphisms of Γ is first-order definable in Γ.*

The second condition in Theorem 19 provides a way of verifying that a given structure is ω-categorical. We illustrate this with the structure $(\mathbb{Q}; <)$. It is not difficult but a good exercise to verify that the orbit of an n-tuple $(t_1, \ldots, t_n)$ from $\mathbb{Q}^n$ with respect to the automorphism group of $(\mathbb{Q}; <)$ is determined by the total quasiorder induced by $(t_1, \ldots, t_n)$ in $(\mathbb{Q}; <)$. We write *total quasiorder*, and not *linear order*, because some of the elements $t_1, \ldots, t_n$ might be equal. There are less than n^n such orders, and hence the automorphism group of $(\mathbb{Q}; <)$ has a finite number of orbits of n-tuples, for all $n \geq 1$. We conclude that $(\mathbb{Q}; <)$ is ω-categorical.

The third condition provides a way of verifying that a structure is not ω-categorical. For example, for every integer $n \geq 1$ there exists a first-order formula $\phi_n(x, y)$ that defines $x = y + n$ over $(\mathbb{Z}; <)$. The formula ϕ_n can be recursively defined as follows.

$$\phi_1(x, y) \quad \Leftrightarrow \quad x < y \wedge \neg \exists w (x < w \wedge w < y)$$

and

$$\phi_n(x, y) \quad \Leftrightarrow \quad \exists z (\phi_{n-1}(x, z) \wedge z < y \wedge \neg \exists w (z < w \wedge w < y))$$

for $n \geq 2$. It follows from Theorem 19 that $(\mathbb{Z}; <)$ is *not* ω-categorical. The fourth condition will become relevant in later sections when we, for instance, discuss canonical Datalog programs in Section 6.3.

There are various ways of constructing ω-categorical structures. One way is to use first-order interpretations and this method will be introduced in Section 3.4. It is also possible to construct ω-categorical structures "from scratch" using *Fraïssé amalgamation*. The details are outside the scope of this survey: Macpherson (2011) outlines the approach while concrete constructions for RCC-5 and RCC-8 have been presented by Bodirsky and Chen (2007) and Bodirsky and Wölfl (2011), respectively. Amalgamation is sometimes called the *patchwork property* in the temporal and spatial reasoning literature (Amaneddine et al., 2013; Huang, 2012; Lutz & Milicic, 2007; Sioutis & Koubarakis, 2012).

3.3 Model-Completeness and Cores

Two structures Γ and Δ might be non-isomorphic and still have the same CSP. This is for instance the case when there simultaneously is a homomorphism from Γ to Δ and a homomorphism from Δ to Γ; this is an immediate consequence of the homomorphism perspective on the CSP (Section 2.1). In this case, we say that Γ and Δ are *homomorphically equivalent*; clearly, this defines an equivalence relation on structures.

Note that there are structures that have the same CSP even when they are not homomorphically equivalent. Consider for example the structures $(\mathbb{Z}; <)$ and $(\mathbb{Q}; <)$. They have the same CSP and there is a homomorphism from $(\mathbb{Z}; <)$ to $(\mathbb{Q}; <)$ but not from $(\mathbb{Q}; <)$ to $(\mathbb{Z}; <)$. The following result implies that for countable ω-categorical structures, homomorphic equivalence captures equality of the CSP.

Lemma 20 (Bodirsky & Dalmau, 2013, Lemma 2)**.** *Let Γ and Δ be structures such that Γ is ω-categorical and Δ is countable. If every finite substructure of Δ homomorphically maps to Γ, then Δ homomorphically maps to Γ.*

We use the lemma as follows. Suppose that Γ and Δ are countable ω-categorical structures such that CSP(Γ) and CSP(Δ) are the same computational problem. Arbitrarily choose a finite substructure Θ of Δ. By viewing CSPs as homomorphism problems as in Section 2.1, Θ can be viewed as an instance of CSP(Δ). There is a trivial homomorphism from Θ to Δ and there is a homomorphism from Θ to Γ since CSP(Δ) and CSP(Γ) are the same computational problem. The lemma above implies that Δ homomorphically maps to Γ. Analogously, there is a homomorphism from Γ to Δ, and this shows that Γ and Δ are homomorphically equivalent.

For ω-categorical structures Γ, much more can be said: the homomorphic equivalence class of Γ contains a distinguished, 'most beautiful' member which is up to isomorphism uniquely given by two properties. One of these properties is fundamental in the study of CSPs over finite domains and in the graph homomorphism literature: it is the concept of *cores*. A relational structure Γ is a *core* if all homomorphisms from Γ to Γ are *embeddings*, that is, they are injective and also preserve the complements of all relations in Γ.

The other property is a central concept in model theory, the concept of *model completeness*. When T is a first-order theory, we say that sentences ϕ and ψ are *equivalent modulo T* if $T \models (\phi \Leftrightarrow \psi)$. A structure Γ is model-complete if its first-order theory T has the following property: every first-order formula is equivalent to an existential formula modulo T.

Theorem 21 (Bodirsky, 2007)**.** *Every ω-categorical structure is homomorphically equivalent to a model-complete core structure Γ which is unique up to isomorphism. Moreover, Γ is ω-categorical and the orbits of n-tuples are primitive positive definable in Γ for all $n \geq 1$.*

Since homomorphically equivalent structures have the same CSP, one can therefore focus on ω-categorical structures that have these properties. Cores are important when studying the complexity of finite-domain CSPs: we refer to Hell and Nešetřil (2004) for a textbook that extensively covers cores in the graph homomorphism literature and to Bulatov, Krokhin and Jeavons (2005) for the relevance of cores for the universal-algebraic approach.

3.4 First-Order Interpretations and Primitive Positive Interpretations

There are two reasons why we introduce first-order interpretations in this survey. Firstly, they are a very flexible tool to construct new ω-categorical structures from known ones. The second reason is

that a specialisation of first-order interpretations, namely *primitive positive interpretations*, are most useful for the study of the computational complexity of CSPs and, as we will see in Section 6.4, also for the question which CSPs can be solved by Datalog and local consistency techniques.

Our presentation of first-order interpretations closely follows Hodges' (1993) book. When $\delta(x_1, \ldots, x_k)$ is a first-order τ-formula with k free variables $x_1, \ldots, x_k$ and Δ is a τ-structure, then we write $\delta(\Delta^k)$ for the k-ary relation that is defined by δ on Δ.

Definition 22. A relational σ-structure Γ has a *(first-order) interpretation* I in a τ-structure Δ if there exists a natural number d, called the *dimension* of I, and

- a τ-formula $\delta_I(x_1, \ldots, x_d)$ – called the *domain formula*,
- for each atomic σ-formula $\phi(y_1, \ldots, y_k)$ a τ-formula $\phi_I(\overline{x}_1, \ldots, \overline{x}_k)$ where the $\overline{x}_i$ denote disjoint d-tuples of distinct variables – called the *defining formulas*,
- a surjective map $h\colon \delta_I(\Delta^d) \to D_\Gamma$ – called the *coordinate map*,

such that for all atomic σ-formulas ϕ and all tuples $\overline{a}_i \in \delta_I(\Delta^d)$

$$\Gamma \models \phi(h(\overline{a}_1), \ldots, h(\overline{a}_k)) \;\Leftrightarrow\; \Delta \models \phi_I(\overline{a}_1, \ldots, \overline{a}_k) \; .$$

If the formulas δ_I and ϕ_I are all primitive positive (Section 2.2), we say that the interpretation I is *primitive positive*. Note that the dimension d, the set $S := \delta(\Delta^k)$, and the coordinate map h determine the defining formulas up to logical equivalence; hence, we sometimes denote an interpretation by $I = (d, S, h)$. We also see that the kernel $\{(a, a') \in A \times A \mid h(a) = h(a')\}$ of h coincides with the relation defined by $(x = y)_I$, for which we also write $=_I$, the defining formula for equality. We have the following two important results.

Lemma 23 (Hodges, 1993, Theorem 7.3.8)**.** *Let Δ be ω-categorical. Then every structure Γ that is first-order interpretable in Δ is ω-categorical.*

Lemma 24 (Bodirsky, 2008, Proposition 3)**.** *Let Γ be a structure with a finite relational signature and a primitive positive interpretation in Δ. Then there is a polynomial-time reduction from* $\mathrm{CSP}(\Gamma)$ *to* $\mathrm{CSP}(\Delta)$.

We will see in Section 6.4 that a variant of Lemma 23 also holds for solvability by Datalog. Another interesting observation is the following: by inspection of the proof that $\mathrm{CSP}(\Gamma)$ and the CSP for the dual encoding Δ of Γ are polynomial-time equivalent (as defined in Proposition 14), one can observe that there is a primitive positive interpretation of Γ in Δ and a primitive positive interpretation of Δ in Γ. Hence, Proposition 14 is a consequence of this observation combined with Lemma 24. It also follows (by Lemma 23) that ω-categoricity is preserved by the dual encoding.

We conclude this section by providing a concrete example of a primitive positive interpretation. For any set B, we define the 6-ary relation I_6^B as follows:

$$\begin{aligned}\{(x_1, x_2, y_1, y_2, z_1, z_2) \in B^6 \;\mid\; & (x_1 = x_2 \wedge y_1 \neq y_2 \wedge z_1 \neq z_2) \vee \\ & (x_1 \neq x_2 \wedge y_1 = y_2 \wedge z_1 \neq z_2) \vee \\ & (x_1 \neq x_2 \wedge y_1 \neq y_2 \wedge z_1 = z_2)\}\end{aligned}$$

Recall Example 3 where we defined the Boolean relation $1\mathrm{IN}3 = \{(0,0,1), (0,1,0), (1,0,0)\}$. We give a primitive positive interpretation I of the structure $(\{0,1\};\ 1\mathrm{IN}3)$ in $(B; I_6^B)$ when

$|B| \geq 2$. This proves, for instance, that $\mathrm{CSP}(B; I_6^B)$ is NP-hard since $\mathrm{CSP}(\{0,1\};\ \mathrm{1IN3})$ is NP-hard. The dimension of I is 2 and the domain formula δ_I is the constant true, i.e. any valid τ-formula. The formula $\mathrm{1IN3}(x_1, x_2, y_1, y_2, z_1, z_2)_I$ is $I_6^B = (x_1, x_2, y_1, y_2, z_1, z_2)$ and the formula $=_I (x_1, x_2, y_1, y_2)$ is

$$\begin{aligned}\exists a_1, a_2, u_1, u_2, u_3, u_4, z_1, z_2 (a_1 = a_2 \wedge I_6^B(a_1, a_2, u_1, u_2, u_3, u_4) \\ \wedge I_6^B(u_1, u_2, x_1, x_2, z_1, z_2) \\ \wedge I_6^B(u_3, u_4, z_1, z_2, y_1, y_2))\end{aligned}$$

One may verify that $=_I (x_1, x_2, y_1, y_2)$ holds if and only if $x_1 = x_2 \Leftrightarrow y_1 = y_2$. Finally, we let the map h act as follows: h maps $(b_1, b_2) \in B^2$ to 1 if $b_1 = b_2$ and to 0 otherwise. Verifying that this is indeed a primitive positive interpretation of $(\{0,1\};\ \mathrm{1IN3})$ in $(B; I_6^B)$ is left to the reader.

3.5 The Universal-Algebraic Approach

We have seen in the previous section that when a structure Γ primitively positively interprets another structure Δ and $\mathrm{CSP}(\Delta)$ is NP-hard, then $\mathrm{CSP}(\Gamma)$ is NP-hard, too. Hence, it is important to understand which structures are primitively positively interpretable in a given structure Γ. For this purpose, universal algebra provides the right tools when Γ is finite, or countably infinite and ω-categorical. The essential notion here is the concept of a *polymorphism* of Γ. The universal-algebraic approach to CSPs has been pioneered by Jeavons and co-authors; we refer to Bulatov, Krokhin, and Jeavons (2005) for an introduction.

Let $R \subseteq D^k$ denote a relation. We say that a function $f\colon D^n \to D$ *preserves* R (or is a *polymorphism* of R) if for arbitrary $t_1, \ldots, t_n \in R$, the tuple

$$(f(t_1[1], \ldots, t_n[1]), f(t_1[2], \ldots, t_n[2]), \ldots, f(t_1[k], \ldots, t_n[k]))$$

is a member of R. A function is a polymorphism of a set of relations Γ if the function preserves every relation in Γ. An *endomorphism* is a unary polymorphism. The set of endomorphisms of a structure forms a monoid with respect to composition. Every automorphism is an endomorphism but an endomorphism is not necessarily an automorphism.

It is known that whether or not a finite structure Γ interprets primitively positively another structure Δ only depends on the polymorphisms of Γ and of Δ. The same is true for ω-categorical structures Γ and Δ (see, e.g., Bodirsky & Pinsker, 2015b). Lemma 24 therefore implies that the computational complexity of the CSP of a finite or countably infinite ω-categorical structure Γ only depends on the polymorphisms of Γ.

By this observation, universal-algebraic techniques can be used to analyse the computational complexity of $\mathrm{CSP}(\Gamma)$. An important feature of the universal-algebraic approach is that tractability of a CSP can be linked to the existence of polymorphisms of the constraint language. This link can be exploited in several directions: first, when we already know that a constraint language of interest has a polymorphism satisfying good properties, then this polymorphism can guide the search for an efficient algorithm for the corresponding CSP. Another direction is when we already have an algorithm (or an algorithmic technique) and we want to know for which CSPs the algorithm is a correct decision procedure: again, polymorphisms are the key tool for this task. Finally, we might use the absence of polymorphisms with good properties to prove that a CSP is NP-hard. One should note that there are *many* instances where the algebraic approach has been successfully applied to

classify the complexity of CSPs with constraint languages over finite domains or ω-categorical constraint languages (Barto & Kozik, 2014; Bodirsky & Kára, 2009; Bodirsky & Pinsker, 2015a; Bulatov, 2006; Jeavons, Cohen, & Gyssens, 1997).

4. What is a Qualitative Formalism?

It is not obvious how to define "qualitative reasoning" rigourously. However, the concept seems to have an informal meaning that is more or less generally accepted. Renz and Nebel (2007, p. 161) write

> Qualitative reasoning is an approach for dealing with commonsense knowledge without using numerical computation. Instead, one tries to represent knowledge using a limited vocabulary such as qualitative relationships between entities or qualitative categories of numerical values, (...)

while Apt and Brand (2006, p. 29) write

> *Qualitative reasoning* was introduced in AI to abstract from numeric quantities, such as the precise time of an event or the location or trajectory of an object in space, and to reason instead on the level of appropriate abstractions.

Problem abstraction is at the heart of both descriptions: qualitative reasoning is about disregarding unnecessary and uninteresting details.

4.1 Partition Schemes

Ligozat and Renz (2004) proposed a formal definition of qualitative formalisms which is inspired by the network satisfaction problem for proper relation algebras. Let D be a non-empty domain. Given a finite family $\mathcal{B} = \{R_1, \ldots, R_k\}$ of binary relations over D, we say that $\mathcal{B}$ is *jointly exhaustive* (JE) if $\bigcup \mathcal{B} = D^2$ and that $\mathcal{B}$ is *pairwise disjoint* (PD) if $R_i \cap R_j = \varnothing$ whenever $1 \leq i \neq j \leq k$. If $\mathcal{B}$ is simultaneously JE and PD, then $\mathcal{B}$ forms a partition of the set D^2. Ligozat and Renz use *partition schemes* as the basis for qualitative formalisms.

Definition 25. Let D be a non-empty domain and let $\mathcal{B} = \{R_1, \ldots, R_k\}$ be a finite set of binary relations over D. We say that $\mathcal{B}$ is a *partition scheme* if the following holds:

1. $\mathcal{B}$ is JEPD,
2. the equality relation $\mathtt{EQ}_D = \{(x, x) \in D^2\}$ is in $\mathcal{B}$, and
3. for every $R_i \in \mathcal{B}$, the converse relation $R_i^\smile$ is in $\mathcal{B}$.

Given a finite set of binary relations $\mathcal{B} = \{R_1, \ldots, R_k\}$, we follow notational conventions (Cohen, Jeavons, Jonsson, & Koubarakis, 2000; Jonsson & Lagerkvist, 2015) and define $\mathcal{B}^{\vee=}$ as the set of all unions of relations from $\mathcal{B}$. Given a partition scheme $\mathcal{B}$, then Ligozat and Renz study the variant of the network satisfaction problem where the labelling function f in the definition of networks (Definition 9) is replaced by a function from V^2 to $\mathcal{B}^{\vee=}$ (and all other definitions that were made in Section 2.3 are adapted accordingly). Thus, the set $\mathcal{B}^{\vee=}$ is the natural object that Ligozat and Renz study.

In the following two subsections, we explain why this definition of a qualitative formalism might be viewed as being both too liberal and too restrictive. This criticism has originally been stated by Westphal, Hué, and Wölfl (2014) and Westphal (2015). We reconsider their line of reasoning and present additional arguments for it. The final subsection of this section (Section 4.4) presents an alternative way of defining qualitative CSPs.

4.2 Partition Schemes: Too Liberal

We will now show that every computational problem can be reduced to a constraint satisfaction problem where the constraint language forms a partition scheme. We start with a simple auxiliary result.

Proposition 26. *Let $\mathcal{R} = \{R_1, \dots, R_k\}$ be a set of binary relations over some domain D. Then there exists a finite set $\mathcal{S}$ of JEPD binary relations with the following properties:*

- *each R_i, for $1 \leq i \leq k$, is a union of relations in $\mathcal{S}$.*
- *each $S \in \mathcal{S}$ is an intersection of relations from $\mathcal{R}$ or complements of relations from $\mathcal{R}$.*

Proof. Let $R_0 := D^2 \setminus \bigcup \mathcal{R}$ and $\mathcal{R}_0 = \{R_0\} \cup \mathcal{R}$. For each subset $X \subseteq \mathcal{R}_0$, define $S_X \subseteq D^2$ such that $t \in S_X$ if and only if t is a member of each relation in X and t is not a member of all relations in $\mathcal{R}_0 \setminus X$. Define $\mathcal{S} = \{S_X \mid X \subseteq \mathcal{R}_0\}$. It is obvious that $\bigcup \mathcal{S} = D^2$, i.e., $\mathcal{S}$ is jointly exhaustive since $R_0 \in \mathcal{R}_0$. It is also obvious that $S \cap S' = \varnothing$ for distinct $S, S' \in \mathcal{S}$, i.e., the relations in $\mathcal{S}$ are pairwise disjoint. Finally, note that for every $R \in \mathcal{R}$

$$R = \bigcup_{S \in \mathcal{S} \text{ and } S \subseteq R} S.$$

□

We continue by extending Proposition 26 to partition schemes.

Proposition 27. *Let $\mathcal{R} = \{R_1, \dots, R_k\}$ be a set of binary relations over some domain D and suppose that one of the following holds:*

1. *$R_i \cap \mathtt{EQ}_D = \varnothing$ for $1 \leq i \leq k$, or*
2. *$\mathtt{EQ}_D \subseteq R_1$ and $R_i \cap \mathtt{EQ}_D = \varnothing$ for $2 \leq i \leq k$.*

Then there exists a finite partition scheme $\mathcal{S}$ such that every relation in $\mathcal{R}$ can be written as a union of relations from $\mathcal{S}$.

Proof. Case 1 is easily reduced to Case 2: add the relation $\mathtt{EQ}_D$ to $\mathcal{R}$. Hence, it is sufficient to consider Case 2. Let $\mathcal{R}_1 := \{\mathtt{EQ}_D, R_0, R_1, \dots, R_k\}$ where $R_0 = D^2 \setminus (\mathtt{EQ}_D \cup \bigcup_{i=1}^{k} R_k)$ and let $\mathcal{R}_2 = \{R, R^\smile \mid R \in \mathcal{R}_1\}$. Let $\mathcal{S}$ be the finite set of relations obtained from applying the construction of the proof of Proposition 26 to $\mathcal{R}_2$. Proposition 26 shows that every relation in $\mathcal{R} \subseteq \mathcal{R}_2$ is the union of relations from $\mathcal{S}$.

1. $\mathcal{S}$ is JEPD. This is stated in Proposition 26.

2. $\mathtt{EQ}_D \in \mathcal{S}$. Note that $\mathtt{EQ}_D \in \mathcal{R}_2$ and $\mathtt{EQ}_D \cap R = \varnothing$ for every $R \in \mathcal{R}_2 \setminus \{\mathtt{EQ}_D\}$. The construction in Proposition 26 clearly implies that $\mathtt{EQ}_D \in \mathcal{S}$, too.

3. If $R \in \mathcal{S}$, then $R^\smile \in \mathcal{S}$. Arbitrarily choose $R \in \mathcal{S}$. By Proposition 26, R is the intersection of relations $R_1, \ldots, R_m$ such that $R_i \in \mathcal{R}_2$ or $D^2 \setminus R_i \in \mathcal{R}_2$. By definition of $\mathcal{R}_2$, the relations $R_1^\smile, \ldots, R_m^\smile$ are in $\mathcal{R}_2$, too, and the construction of $\mathcal{S}$ shows that $R^\smile = R_1^\smile \cap \cdots \cap R_m^\smile$ is in $\mathcal{S}$.

Hence, $\mathcal{S}$ is a partition scheme and the proof is complete. □

It is now easy to translate arbitrary computational problems into CSPs for partition schemes.

Proposition 28. *Let L be a language over a finite alphabet $\Sigma = \{0, \ldots, m\}$. Then there is a structure Γ with finite relational signature whose relations form a partition scheme such that L is polynomial-time Turing reducible to* $\mathrm{CSP}(\Gamma)$.

Proof. Bodirsky and Grohe (2008, Thm. 1) show that there exists a relational structure Γ_1 over an infinite domain D such that L and $\mathrm{CSP}(\Gamma_1)$ are equivalent under polynomial-time Turing reductions. The relational structure Γ_1 contains one unary relation P_a for each $a \in \Sigma$, one binary relation N, and two unary relations S and T. By analysing the construction used in Bodirsky and Grohe's proof, one may verify that $(x, x) \notin N$ for arbitrary $x \in D$. Introduce a dummy element d such that $d \notin D$. Given a unary relation $U \subseteq D$, we define the binary relation $\hat{U} = U \times \{d\}$, i.e. we make it binary by extending it with a "useless" argument. Let $\Gamma_2 = \{\hat{P}_0, \ldots, \hat{P}_m, N, \hat{S}, \hat{T}\}$ be a relational structure with domain $D \cup \{d\}$ and note that (1) $\mathrm{CSP}(\Gamma_1)$ and $\mathrm{CSP}(\Gamma_2)$ are polynomial-time equivalent, and (2) the relations $\mathcal{R}$ in Γ_2 satisfy the conditions of Case 1 in Proposition 27, which then implies the result. □

Proposition 28 implies that in particular the *satisfiability problem of polynomial inequalities over the reals* (SPPI) can be reduced to the CSP of a binary structure whose relations form a partition scheme. SPPI can be formulated as $\mathrm{CSP}(\mathbb{R}; <, R_+, R_*, R_1)$ where $<$ is the usual order of the real numbers, $R_+ := \{(x, y, z) \mid x + y = z\}$, $R_* := \{(x, y, z) \mid x \cdot y = z\}$, and $R_1 := \{1\}$. This problem is often viewed as the archetypical *quantitative* problem, since it allows (for example) to reason about absolute points, distances, and elementary arithmetic. For results about the computational complexity of SPPI, we refer to the paper by Schaefer (2009).

Proposition 28 only implies the existence of a polynomial-time reduction from SPPI to the CSP for a partition scheme. But in fact, we can also come up with a partition scheme where the relationship with SPPI is much tighter, as we will see in the following. This indicates that Ligozat and Renz's approach fails to capture *qualitative* reasoning problems in the sense outlined in the beginning of this section, since it allows to model one of the most expressive and genuinely *quantitative* formalisms.

Theorem 29. *There exists a partition scheme $\mathcal{B}$ of binary relations over a set D such that the following two structures are mutually primitive positive interpretable.*

1. $(\mathbb{R}; <, R_+, R_*, R_1)$
2. *The binary structure Γ with domain D whose set of relations is $\mathcal{B}^{\vee =}$.*

In particular, the CSPs for these two structures are polynomial-time equivalent.

Proof. We use $D := \mathbb{R}^3$, and consider the following five binary relations:

$$\begin{array}{rcl} \mathrm{Less} & := & \{((a,b,p),(c,d,q)) \subseteq (\mathbb{R}^3)^2 \mid a < c \wedge p \neq q\} \\ \mathrm{Add} & := & \{((a,b,p),(c,d,q)) \subseteq (\mathbb{R}^3)^2 \mid a + b = c \wedge p \neq q\} \\ \mathrm{Mult} & := & \{((a,b,p),(c,d,q)) \subseteq (\mathbb{R}^3)^2 \mid a \cdot b = c \wedge p \neq q\} \\ \mathrm{One} & := & \{((a,b,p),(c,d,q)) \subseteq (\mathbb{R}^3)^2 \mid a = 1 \wedge p \neq q\} \\ \mathrm{Link} & := & \{((a,b,p),(c,d,q)) \subseteq (\mathbb{R}^3)^2 \mid b = c \wedge p \neq q\} \end{array}$$

Let $\mathcal{R} := \{\mathrm{Less}, \mathrm{Add}, \mathrm{Mult}, \mathrm{One}, \mathrm{Link}\}$. The idea here is to use the first coordinate for representing real numbers, the second coordinate as a storage, and the third argument for avoiding the equality relation: note that Proposition 27 is applicable to $\mathcal{R}$ since tuples of the form (x, x) do not appear in these relations. Let $\mathcal{B}$ be the partition scheme constructed for $\mathcal{R}$ in the proof of Proposition 27, and let Γ be a binary structure with the domain $\mathbb{R}^3$ and the relations from $\mathcal{B}^{\vee=}$.

We start with a primitive positive interpretation of Γ in $(\mathbb{R}; <, R_+, R_*, R_1)$. The interpretation is 2-dimensional and the domain formula $\delta(x, y)$ is $x = x \wedge y = y$. The relations in $\mathcal{R}$ are obviously first-order definable in $(\mathbb{R}; <, R_+, R_*, R_1)$, and the construction of the relations in $\mathcal{B}$ from the proof of Proposition 27 is first-order, too. It is then clear that relations from $\mathcal{B}^{\vee=}$ are also first-order.

We will show that every first-order formula (recall that we do not allow constants from $\mathbb{R}$ in such formulas) is in $(\mathbb{R}; <, R_+, R_*, R_1)$ equivalent to a primitive positive formula. First note that

$$(x = y \vee u = v) \Leftrightarrow (x - y)(u - v) = 0 \,.$$

Moreover,

$$\begin{aligned} \phi(x,y,z) \vee \phi'(x,y,z) \Leftrightarrow \exists u, v, w, u', v', w' \big(& \phi(u,v,w) \wedge \phi'(u',v',w') \\ & \wedge \ ((x = u \wedge y = v \wedge z = w) \vee (x = u' \wedge y = v' \wedge z = w')) \big) \end{aligned}$$

Using this idea, one can show that finite disjunctions are primitive positive definable in the language of $(\mathbb{R}; <, R_+, R_*, R_1)$. Then note that atomic negation can be eliminated using disjunction: $\neg(x < y)$ is equivalent to $x = y \vee y < x$, $\neg(x = y)$ is equivalent to $x < y \vee y < x$, $\neg R_+(x, y, z)$ is equivalent to $\exists z'(R_+(x, y, z') \wedge z' \neq z)$, and similarly for $\neg R_*$. By the famous theorem of Tarski and Seidenberg (cf. Goodman & O'Rourke, 2004, Sec. 33), the structure $(\mathbb{R}; <, R_+, R_*, R_1)$ has quantifier elimination when we allow constants from $\mathbb{R}$ in the formulas. It is well-known that when the first-order formula we started with does not have any constants, then the formula produced by Tarski and Seidenberg will only have algebraic constants, and those have primitive positive definitions in $(\mathbb{R}; <, R_+, R_*, R_1)$ as well.

The interpretation of $(\mathbb{R}; <, R_+, R_*, R_1)$ in Γ is 1-dimensional, the domain formula $\delta(x)$ is $x = x$, and the coordinate map $h \colon \mathbb{R}^3 \to \mathbb{R}$ is defined by $h(a, b, c) := a$. Note that

$$\begin{array}{rcl} h(x) = 1 & \text{iff} & \exists u.\, \mathrm{One}(x, u) \\ h(x) < h(y) & \text{iff} & \exists u \big(\mathrm{Less}(x, u) \wedge \mathrm{Less}(u, y) \big) \\ h(x) = h(y) & \text{iff} & \exists u \big(\mathrm{Link}(u, x) \wedge \mathrm{Link}(u, y) \big) \\ h(z) = h(x) + h(y) & \text{iff} & \exists u \big(\mathrm{Add}(u, z) \wedge \mathrm{Link}(x, u) \wedge \mathrm{Link}(u, y) \big) \\ h(z) = h(x) \cdot h(y) & \text{iff} & \exists u \big(\mathrm{Mult}(u, z) \wedge \mathrm{Link}(x, u) \wedge \mathrm{Link}(u, y) \big) \end{array}$$

Since all these formulas are primitive positive over the signature of Γ, the statement follows. □

4.3 Partition Schemes: Too Restrictive

The restriction to binary structures that is underlying the definition of Ligozat and Renz is somewhat artificial. One argument for binary formalisms has been that most formalisms in the literature are binary. On the other hand, there are also many interesting (qualitative and quantitative) formalisms for constraints of higher arity. One can, in principle, always use the dual encoding but the resulting CSPs may be difficult to work with.

Another argument is that several reasoning techniques (most importantly consistency methods) are tailored towards binary formalisms. As we will see in the forthcoming Section 6 on Datalog, the consistency methods can be generalised in a both natural and systematic way so that they work for arbitrary finite structures.

We believe that there is another aspect why the proposal of Ligozat and Renz is too restrictive: the restriction to *unions* of JEPD relations. There are binary structures with natural CSPs whose relations happen not to be of this form, like the polynomial-time tractable CSP for the *basic relations* in Allen's interval algebra, which are not closed under unions. In fact, closing the set of relations under unions leads to the full Allen algebra, which has an NP-complete CSP. Thus, we claim that the study of formalisms that are not closed under unions is important, but these formalisms do not satisfy the definition of qualitative formalisms proposed by Ligozat and Renz. Insisting on closure under unions is also not in the spirit of the initial intuitive meaning of qualitative formalisms.

4.4 Qualitative CSPs and ω-Categoricity

We propose the following definition of what it means for a CSP to be qualitative. As we will see at the end of this section, it captures some of the aspects of the intuitive notion of a qualitative formalism: abstracting away unnecessary details and the impossibility of simulating arithmetic.

Definition 30. Let Γ be a structure with finite relational signature. We say that $\mathrm{CSP}(\Gamma)$ is *qualitative* if there exists an ω-categorical structure Δ such that $\mathrm{CSP}(\Gamma)$ and $\mathrm{CSP}(\Delta)$ are the same computational problem.

Note that we do *not* require that Γ itself is ω-categorical; we merely require that there *exists* an ω-categorical structure which has the same CSP. As we will see, this subtle difference is important. Consider for example $\Gamma = (\mathbb{Z}; <)$. We have seen before that Γ is not ω-categorical. However, $\mathrm{CSP}(\Gamma)$ equals $\mathrm{CSP}(\mathbb{Q}; <)$, and $(\mathbb{Q}; <)$ *is* ω-categorical.

The same issue holds for RCC-5. The structure RCC-5_{set} presented in Section 2.5 and also the structures of the representations of RCC-5 presented there are *not* ω-categorical: one can find for each $n \geq 1$ a first-order formula $\phi_n(x, y)$ that holds if and only if $x\{\mathtt{PP}\}y$ and $|y| = |x| + n$ (in a way that is analogous to the construction presented after Theorem 19) and then apply Theorem 19. However, there do exist ω-categorical structures that have the same CSP as RCC-5_{set} (and this will be explained in Section 5.3). Hence, RCC-5 is a qualitative formalism according to our proposal, and this coincides with the intuitive definition given earlier. Let us also mention that the standard representation of Allen's Interval Algebra is an ω-categorical structure – see Section 5.2.

It is now interesting to go back to the SPPI problem and the structure Γ that was introduced in Section 4.2: is it qualitative or not according to our definition? This raises an immediate question: how to verify that a given structure Δ admits an ω-categorical structure Γ that has the same CSP? There is a result by Bodirsky, Hils, and Martin (2010) that might help to answer this question. The

result shows that such structures Δ can be characterised by a finiteness condition that resembles the classical condition of a language to be regular given by Myhill and Nerode (1979, Sec. 3).

Definition 31. Let Γ be any structure with relational signature τ, and let ϕ_1 and ϕ_2 be primitive positive τ-formulas with the same free variables $x_1, \ldots, x_n$. Define $\phi_1 \sim_n \phi_2$ if for all primitive positive τ-formulas ψ with free variables $x_1, \ldots, x_n$, the instance $\exists x_1, \ldots, x_n(\phi_1 \wedge \psi)$ of $\mathrm{CSP}(\Gamma)$ is satisfiable if and only if $\exists x_1, \ldots, x_n(\phi_2 \wedge \psi)$ is satisfiable.

Clearly, $\sim_n$ is for all $n \in \mathbb{N}$ an equivalence relation. We say that an equivalence relation has *finite index* if and only if it contains a finite number of equivalence classes.

Theorem 32 (Bodirsky, Hils, and Martin, 2010)**.** *Let Γ be any relational structure. Then there exists an ω-categorical structure Δ which has the same* CSP *as Γ if and only if the relation $\sim_n$ has finite index for all $n \in \mathbb{N}$.*

The forward direction of Theorem 32 is an immediate consequence of Theorem 19. The backward direction is more interesting, and we just mention that it involves the concept of *existential positive completion* and a back-and-forth argument.

Consider for example the structure $(\mathbb{Z}; <)$. Over this structure the primitive positive formulas

$$\exists y_1, \ldots, y_k \; (x_1 < y_1 < y_2 < \cdots < y_k < x_2)$$

with the free variables x_1, x_2 are for all $k \in \mathbb{N}$ inequivalent. However, all of those formulas are equivalent under $\sim_2$. And indeed, as mentioned above, the computational problem $\mathrm{CSP}(\mathbb{Z}; <)$ can also be formulated with the ω-categorical template $\mathrm{CSP}(\mathbb{Q}; <)$.

Let us now return to the SPPI problem and the structure Γ. Needless to say, this structure is *not* ω-categorical. For each $i \in \mathbb{N}$, consider the unary relation

$$U_i := \{(a, b, c) \in \mathbb{R}^3 \mid a = i\}.$$

Clearly, U_1 is primitive positive definable in Γ by $U_1(x) = \exists x'.\ \mathrm{One}(x, x')$. Recall (from Section 4.2) that for arbitrary $(a_1, a_2, a_3) \in \mathbb{R}^3$ we define $h(a_1, a_2, a_3) := a_1$ and that there is a primitive positive definition of $h(z_1, z_2, z_3) = h(x_1, x_2, x_3) + h(y_1, y_2, y_3)$. Thus, we can define U_2 in Γ as follows:

$$U_2(x) = \exists y \, (U_1(y) \wedge h(x) = h(y) + h(y))$$

The relations $U_4, U_8, \ldots$ can be defined similarly. All the unary relations $U_1, U_2, U_4, \ldots$ are in distinct orbits (by Proposition 16), so Theorem 19 implies that Γ is not ω-categorical. To show that $\mathrm{CSP}(\Gamma)$ is not qualitative in our sense, we have to show something stronger: we have to show that this CSP cannot be formulated with an ω-categorical template (which may be different from Γ). But it is easy to see that the the defining primitive positive formulas for the relations U_i are inequivalent with respect to $\sim_1$. Hence, Theorem 32 implies that the CSP cannot be formulated with an ω-categorical template. Therefore, the structures Γ and $\Gamma^{\vee=}$ are *not* qualitative in our sense (and this matches the intuitive meaning of qualitative formalisms from the beginning of this section).

We claim that our definition of a qualitative formalism captures two important aspects of the intuitive notion:

1. Abstracting away unnecessary details: if two solutions f and g of an instance of a CSP are such that there exists an automorphism α of the constraint language with $f = \alpha \circ g$, then f and g share all the essential properties. In this way, the set of all solutions are grouped in equivalence classes that correspond to qualitative abstractions of solutions. When Γ is ω-categorical, then for each number of variables, there are only finitely many such classes to consider since the automorphism group of Γ is oligomorphic.

2. The impossibility of simulating arithmetic: constraint languages that can express arithmetic constraints, as demonstrated for SPPI constraints, do not satisfy the condition given in Theorem 32, and hence are not qualitative according to our definition.

5. Relation Algebras and ω-Categoricity

We have seen (in Section 2) the definition of CSPs as homomorphism problems, how to match the terminology from CSPs with the network satisfaction terminology in relational algebra, and that every network satisfaction problem and also every general network satisfaction problem corresponds to a CSP. However, our RCC-5 examples in Section 2.5.2 show that relation algebras and CSPs are not a perfect match even if we restrict ourselves to binary signatures. We have to be careful in (at least) two respects:

1. different representations of the same relation algebra may have different CSPs and

2. a binary structure whose CSP equals the general network satisfaction problem for a relation algebra $\mathbf{A}$ need not be a representation of $\mathbf{A}$.

In response to the first item, we know that there is always a representation whose network satisfaction problem equals the general network satisfaction problem (Proposition 13).

The second item is more interesting and it has led some researchers to weaken the definition of proper relation algebras on a domain D as we already have seen in Section 4. Many partition schemes of interest (such as RCC-5_{set}) fail to be proper relation algebras since, in particular, the composition of relations is not always a member of the partition scheme. In order to formalise some form of inference despite this, Renz and Ligozat (2005) suggest the use of *weak* composition where ordinary composition is replaced by the largest relation in $\mathcal{B}$ containing ordinary composition. That is, the weak composition $R \diamond S$ of R and S over a set $\mathcal{B}$ of JEPD relations is the relation $\bigcup\{T \in \mathcal{B} \mid (R \circ S) \cap T \neq \varnothing\}$. The introduction of the weak composition operator has had a strong impact on qualitative CSP research and it is routinely used in many different contexts. We will show that weak composition is *not* necessary when working with binary ω-categorical structures: there is always a suitable relation algebra (which we call the *orbital relation algebra*) where we can use ordinary composition.

5.1 Orbital Relation Algebras

We will now describe how every ω-categorical structure Γ naturally gives rise to a certain finite relation algebra $\mathbf{A}$. If Γ is binary, then $\mathrm{CSP}(\Gamma)$ corresponds to the network satisfaction problem for this interpretation of $\mathbf{A}$ restricted to some subset of the relations of $\mathbf{A}$. Since $\mathbf{A}$ is a relation algebra, we do not have to work with weak composition and this shows that there are clear advantages of working with ω-categorical structures whenever possible.

Let G be a permutation group. For arbitrary $a, b \in D$, the *orbital* of (a, b) in G is the set $\{(\alpha(a), \alpha(b)) \mid \alpha \in G\}$, that is, the orbit of the pair (a, b).

Proposition 33. *Let Γ be a structure such that $\mathrm{Aut}(\Gamma)$ has finitely many orbitals. Then the unions of these orbitals form a finite proper relation algebra.*

Proof. Clearly, the orbitals of $\mathrm{Aut}(\Gamma)$ are JEPD. Every binary relation with a first-order definition in Γ is preserved by all automorphisms of Γ (Proposition 16) and hence is a union of orbitals of $\mathrm{Aut}(\Gamma)$. Since composition is first-order definable, it follows that unions of orbitals of $\mathrm{Aut}(\Gamma)$ are preserved under composition. Also the other properties of proper relation algebras in Definition 5 are straightforward to verify. □

Proposition 33 implies in particular that the set of orbitals of $\mathrm{Aut}(\Gamma)$ is JEPD. We call the relation algebra described in Proposition 33 the *orbital relation algebra* of Γ.

Corollary 34. *Every ω-categorical structure has a finite orbital relation algebra.*

Proof. This is an immediate consequence of Theorem 19. □

If we additionally assume that Γ is a model-complete core (as defined in Section 3.3), then the orbitals are even primitive positive definable (by Theorem 21) and the computational complexity does not change if we add them to the structure Γ (by Lemma 24 or the discussion concerning primitive positive definability in Section 2.2). In the forthcoming two sections, we will analyse the orbital relation algebras that arise from Allen's Interval Algebra and RCC-5.

5.2 Example I: Allen's Interval Algebra

Recall the standard representation of Allen's Interval algebra over the rationals that we presented in Section 2.5. Viewed as a relational structure, the domain is $\mathbb{I}$ and we have a relational signature τ with $2^{13} = 8192$ binary relation symbols for each of the element of the relation algebra, with the interpretations as given in Section 2.5.

The ω-categoricity of this structure has already been observed by Hirsch (1996) and this is an easy consequence of Theorem 19 since the number of orbits of n-tuples of the structure with domain $\mathbb{I}$ is bounded by $(2n)^{2n}$ (see the discussion following Theorem 19). The orbital relation algebra for this structure (as introduced in Section 5) is precisely Allen's Interval Algebra: to see this, first observe that pairs of intervals that lie in different base relations must lie in different orbitals since automorphisms must preserve first-order definable relations (Proposition 16). Conversely, when two pairs of intervals

$$([I_1^-, I_1^+], [I_2^-, I_2^+]), ([J_1^-, J_1^+], [J_2^-, J_2^+])$$

lie in the same base relation, then the map that sends I_1^- to J_1^-, I_2^-, J_2^-, I_1^+ to J_1^+, and I_2^+ to J_2^+ is well-defined and preserves $<$ (by inspection of the 13 base relations). Since the orbit of a finite tuple under $\mathrm{Aut}(\mathbb{Q}; <)$ is given by the total quasiorder induced by $<$ on the entries of the tuple (Section 3.2), this means that this partial map can be extended to an automorphism α of $(\mathbb{Q}; <)$. But α applied componentwise to intervals is an automorphism of the structure from the standard representation of Allen's Interval Algebra and maps I_1 to J_1 and I_2 to J_2; therefore (I_1, I_2) and (J_1, J_2) lie in the same orbital.

5.3 Example II: RCC-5

Just as Allen's Interval Algebra, the algebra RCC-5 has an ω-categorical representation[4] (Bodirsky & Chen, 2009). The representation constructed by Bodirsky and Chen is based on *Fraïssé's amalgamation method* (Hodges, 1993) and has the property that the network satisfaction problem for the representation equals the general network satisfaction problem for RCC-5.

We present a different ω-categorical representation with the same property, but using a first-order interpretation in a known ω-categorical structure instead of the amalgamation method. The known ω-categorical structure is the *countable atomless Boolean algebra* $(\mathbb{A}; \cap, \cup, c, \mathbf{0}, \mathbf{1})$ (Hodges, 1993). A Boolean algebra is called *atomless* if it does not contain any atoms, i.e. elements $x \neq \mathbf{0}$ such that for all y with $x \cap y = y$ and $x \neq y$ we have $y = \mathbf{0}$. Note that $(\mathcal{P}(\mathbb{N}); \cup, \cap, c, \varnothing, \mathbb{N})$ is not atomless since every singleton set is an atom. We show, for illustrative purposes, how to construct a countable atomless Boolean algebra. Consider the subsets of $\mathbb{N}$. Every such subset S can be viewed as an infinite sequence of 0s and 1s: if the symbol on the i-th position is 1 then $i \in S$ and $i \notin S$ otherwise. Given this representation, we see, for instance, that the operation $\cap$ is the pointwise minimum of two sequences while the operation $\cup$ is the pointwise maximum of two sequences. We say that a sequence $(x_i)_{i\in\mathbb{N}} = x_0, x_1, x_2, \ldots$ is *periodic* if and only if there exists some number $n > 0$ such that $x_i = x_{i+n}$ for all $i \geq 0$. We say that $(x_i)_{i\in\mathbb{N}}$ has *periodicity* n if n is the smallest number witnessing the periodicity of $(x_i)_{i\in\mathbb{N}}$. Let B contain all sets represented by periodic sequences. One may easily verify that B indeed induces a Boolean algebra. For instance, let $(x_i)_{i\in\mathbb{N}}$ be a sequence with periodicity m and $(y_i)_{i\in\mathbb{N}}$ a sequence with periodicity n. The sequence $(\max(x_i, y_i))_{i\in\mathbb{N}}$ has periodicity smaller or equal than the least common multiple of m and n. It is also clear that B is atomless: for any sequence $(x_i)_{i\in\mathbb{N}}$ that is different from $\mathbf{0} = (0)_{i\in\mathbb{N}}$, there exists a non-zero sequence $(y_i)_{i\in\mathbb{N}} \in B$ such that $(\min(x_i, y_i))_{i\in\mathbb{N}} \neq \mathbf{0}$ and $(y_i)_{i\in\mathbb{N}} \neq (x_i)_{i\in\mathbb{N}}$: let $\bar{y} := x_0 \ldots x_{n-1}0^n x_0 \ldots x_{n-1}0^n \ldots$ where 0^n denotes a sequence of zeroes of length n.

It is well-known that all countable atomless Boolean algebras are isomorphic (Koppelberg, 1989, Corollary 5.16). Since the axioms of Boolean algebras and the property of not having atoms can all be written as first-order sentences, it follows that this structure is ω-categorical. Let $(\mathbb{A}; \cap, \cup, c, \mathbf{0}, \mathbf{1})$ denote such a countable atomless Boolean algebra. It is also well-known that this algebra is *homogeneous* in the sense that isomorphisms between finite subalgebras can be extended to automorphisms (Hodges, 1993, Example 4 on page 100).

The idea how to obtain a representation of RCC-5 using the atomless Boolean algebra is hinted at by Düntsch, Wang, and McCloskey (2001, Proposition 4.4). Formally, let $\mathtt{PP}$, $\mathtt{PP}^\smile$, $\mathtt{DR}$, $\mathtt{PO}$, $\mathtt{EQ}$ be the binary relations with the following first-order definition in $(\mathbb{A}; \cap, \cup, c, \mathbf{0}, \mathbf{1})$.

$$
\begin{aligned}
\mathtt{PP}(x, y) &\text{ iff } (x \cap y = x) \wedge x \neq y \wedge x, y \notin \{\mathbf{0}, \mathbf{1}\} \\
\mathtt{PP}^\smile(x, y) &\text{ iff } (x \cap y = y) \wedge x \neq y \wedge x, y \notin \{\mathbf{0}, \mathbf{1}\} \\
\mathtt{DR}(x, y) &\text{ iff } (x \cap y = \mathbf{0}) \wedge x \neq y \wedge x, y \notin \{\mathbf{0}, \mathbf{1}\} \\
\mathtt{PO}(x, y) &\text{ iff } \neg\mathtt{DR}(x, y) \wedge \neg\mathtt{PP}(x, y) \wedge \neg\mathtt{PP}^\smile(x, y) \wedge x \neq y \wedge x, y \notin \{\mathbf{0}, \mathbf{1}\} \\
\mathtt{EQ}(x, y) &\text{ iff } x = y \wedge x, y \notin \{\mathbf{0}, \mathbf{1}\}
\end{aligned}
$$

The structure $(\mathbb{A} \setminus \{\mathbf{0}, \mathbf{1}\}; \mathtt{PP}, \mathtt{PP}^\smile, \mathtt{DR}, \mathtt{PO}, \mathtt{EQ})$ has a (1-dimensional) first-order interpretation in $(\mathbb{A}, \cap, \cup, c, \mathbf{0}, \mathbf{1})$ and it is consequently ω-categorical by Lemma 23. The orbital algebra of this

4. The same is true for RCC-8 (Bodirsky & Woelfl, 2011).

structure does not equal RCC-5, but rather the relation algebra commonly referred to as RCC-7, as pointed out by Düntsch (2001, Sec. 4.4). In order to obtain a representation of RCC-5, we consider the ω-categorical model-complete core of this structure, which exists by Theorem 21, and which we denote by RCC-$5_{\omega\text{-cat}}$.

Proposition 35. *The orbital relation algebra of RCC-$5_{\omega\text{-cat}}$ equals RCC-5. The network satisfaction problem for this representation of RCC-5 equals the general network satisfaction problem for RCC-5.*

Proof. We have to show that the orbitals of $\mathrm{Aut}(\text{RCC-}5_{\omega\text{-cat}})$ are precisely the five relations denoted by $\mathtt{PP}, \mathtt{PP}^\smile, \mathtt{DR}, \mathtt{PO}, \mathtt{EQ}$ in RCC-$5_{\omega\text{-cat}}$. Recall from Theorem 21 that the orbitals in RCC-$5_{\omega\text{-cat}}$ are primitive positive definable. Hence, it suffices to show that whenever $\phi_1(x, y)$ and $\phi_2(x, y)$ are such primitive positive definitions of orbitals that are contained in the same relation $R \in \{\mathtt{PP}, \mathtt{PP}^\smile, \mathtt{DR}, \mathtt{PO}, \mathtt{EQ}\}$, then $\phi_1(x, y) \wedge \phi_2(x, y)$ is satisfiable in RCC-$5_{\omega\text{-cat}}$ (and hence, the two orbitals are the same). The primitive positive formula $\phi_1(x, y) \wedge \phi_2(x, y)$ is satisfiable in RCC-$5_{\omega\text{-cat}}$ if and only if it is satisfiable in $\Gamma := (\mathbb{A} \setminus \{\mathbf{0}, \mathbf{1}\}; \mathtt{PP}, \mathtt{PP}^\smile, \mathtt{DR}, \mathtt{PO}, \mathtt{EQ})$. Let $a_1, b_1, a_2, b_2 \in \mathbb{A} \setminus \{\mathbf{0}, \mathbf{1}\}$ be such that $\phi_1(a_1, b_1)$ and $\phi_2(a_2, b_2)$ hold in Γ. If $(a_1, b_1), (a_2, b_2) \in R$ and $R \in \{\mathtt{PP}, \mathtt{PP}^\smile, \mathtt{EQ}\}$, then by the homogeneity of $(\mathbb{A}; \cap, \cup, c, \mathbf{0}, \mathbf{1})$ there is an automorphism of $(\mathbb{A}, \cap, \cup, c, \mathbf{0}, \mathbf{1})$ that maps (a_1, b_1) to (a_2, b_2) and, consequently, $\phi_1(x, y) \wedge \phi_2(x, y)$ is satisfiable in Γ. If $R \in \{\mathtt{DR}, \mathtt{PO}\}$, then there might not be such an automorphism of $(\mathbb{A}, \cap, \cup, c, \mathbf{0}, \mathbf{1})$ since it might e.g. be the case that $a_1 \cup b_1 = \mathbf{1}$ and $a_2 \cup b_2 \neq \mathbf{1}$. Let $\psi_1(x, y)$ and $\psi_2(x, y)$ be the formulas obtained from $\phi_1(x, y)$ and $\phi_2(x, y)$ by replacing all relation symbols from Γ by their definition over the Boolean algebra $(\mathbb{A}; \cap, \cup, c, \mathbf{0}, \mathbf{1})$. It follows from the proof of Proposition 15 that for $i \in \{1, 2\}$ there exists a Boolean algebra A_i and elements a_i, b_i of A_i such that

$$a_i \cup b_i \neq \mathbf{1} \wedge a_i \neq \mathbf{0} \neq b_i \wedge a_i \neq \mathbf{0} \neq b_i \wedge \psi_i(a_i, b_i) \tag{1}$$

holds in A_i. It is well-known (Marriott & Odersky, 1996, Corollary 5.6) that every existential formula in the language of Boolean algebras that is satisfiable in some Boolean algebra is satisfiable in all infinite Boolean algebras, and in particular in $(\mathbb{A}; \cap, \cup, c, \mathbf{0}, \mathbf{1})$. Let a_i', b_i' be the elements that witness that (1) is satisfiable in $(\mathbb{A}; \cap, \cup, c, \mathbf{0}, \mathbf{1})$. Then there exists an automorphism α of $(\mathbb{A}, \cap, \cup, c, \mathbf{0}, \mathbf{1})$ that maps (a_1', b_1') to (a_2', b_2'). Again, we obtain that $\psi_1(x, y) \wedge \psi_2(x, y)$ is satisfiable in $(\mathbb{A}; \cap, \cup, c, \mathbf{0}, \mathbf{1})$, and hence $\phi_1(x, y) \wedge \phi_2(x, y)$ is satisfiable in RCC-$5_{\omega\text{-cat}}$.

To show that the network satisfaction problem the representation of RCC-5 obtained from RCC-$5_{\omega\text{-cat}}$ equals the general network satisfaction problem for RCC-5, we have to show that every RCC-5 network N that is satisfiable in *some* representation of RCC-5 is also satisfiable with respect to this representation. Proposition 15 shows that ϕ_N is satisfiable in RCC-5_{set}. The fact by Marriott and Odersky quoted above implies that ϕ_N is satisfiable in $(\mathbb{A} \setminus \{\mathbf{0}, \mathbf{1}\}; \mathtt{PP}, \mathtt{PP}^\smile, \mathtt{DR}, \mathtt{PO}, \mathtt{EQ})$, and hence in RCC-$5_{\omega\text{-cat}}$ which has the same CSP. This concludes the proof. □

6. Datalog

We will now consider the connections between Datalog and ω-categorical CSPs. Datalog is an important algorithmic tool for obtaining polynomial-time algorithms and it has been used in many different contexts. The main application has historically been in databases (Ramakrishnan & Ullman, 1995) but the range of applications has recently broadened significantly (Huang, Green, & Loo, 2011). For those intrested in an accessible introduction to Datalog, we recommend the survey

by Ceri, Gottlob, and Tanca (1989). Datalog can be viewed as the language of logic programs without function symbols (Kolaitis & Vardi, 1998; Ebbinghaus & Flum, 1999). For those who prefer a database oriented description of Datalog, one can view Datalog as conjunctive queries extended with a mechanism for recursion, cf. Abiteboul, Hull, and Vianu (1995) or the textbook by Nilsson and Małuszyński (1990).

For constraint satisfaction with finite domains, Datalog was first investigated systematically by Feder and Vardi (1999). Many local consistency procedures for finite-domain CSPs can be directly formulated with Datalog programs (we discuss this later for the path consistency procedure). Note that we do not claim that every local consistency algorithm can be formulated with Datalog. However, for the known local consistency algorithms it is true that if a CSP can be solved by this algorithm, then it can also be solved by a Datalog program (of course, this cannot be made a formal statement since there is no universally accepted formal definition of local consistency).

The finite-domain CSPs having bounded width are fully known due to results by Barto and Kozik (2014). Note that all known tractable finite-domain CSPs are solvable by a combination of two algorithmic principles: Datalog solvability and algorithms based on "the few subpowers property" (Idziak, Markovic, McKenzie, Valeriote, & Willard, 2010). Most people would agree that algorithms based on the few subpowers property should not be considered local consistency algorithms. On a very abstract level, they are similar to algorithms for solving linear equation systems (such as Gaussian elimination) and are thus concerned by global properties of the given CSP instance.

For CSPs with infinite domains, Datalog programs play an important role even though this is usually not visible: path-consistency checking (and k-consistency in general) can easily be formulated in Datalog. Moreover, Datalog can readily be used for defining consistency procedures that generalise the ordinary consistency checking procedures in order to handle higher-arity constraints, for example. As expected, there are many choices of constraint languages Γ such that there does not exist *any* Datalog program Π solving $\mathrm{CSP}(\Gamma)$. This is obviously true when $\mathrm{CSP}(\Gamma)$ is not polynomial-time solvable but there are also examples of Γ such that $\mathrm{CSP}(\Gamma)$ is polynomial-time solvable but $\mathrm{CSP}(\Gamma)$ cannot be solved by Datalog. We will not discuss such results in any depth (though we will come back to them in Section 7) but we note that they are known to be quite difficult to obtain, and the few existing results often utilise interesting combinatorial results (Bodirsky & Mueller, 2011; Bodirsky & Kára, 2010; Feder & Vardi, 1999; Kolaitis & Vardi, 1995).

Clearly, there may be a very large number of local consistency methods and Datalog-based methods for infinite-domain CSPs. The situation seems simpler in the case of ω-categorical constraint languages where there is an intriguing possibility based on *canonical Datalog programs*. In short, every Datalog program can be assigned a *width tuple* (l, k) that, in a certain sense, measures the complexity of the program. Bodirsky and Dalmau (2013) have shown that for every pair $(l, k) \in \mathbb{N}^2$ there exists a *canonical Datalog program* Π *of width* (l, k) (defined in Section 6.3) with the property that $\mathrm{CSP}(\Gamma)$ can be solved by a Datalog program of width (l, k) if and only if it can be solved by Π. Thus, one may concentrate on canonical Datalog programs when studying ω-categorical CSPs.

The basics of the Datalog language will be introduced in Section 6.1 and we discuss connections with path-consistency (and other notions of consistency) in Section 6.2. After this, canonical Datalog programs will be introduced in Section 6.3. Solvability by Datalog is a mathematically robust concept: for example, it is preserved by primitive positive interpretability and this will be discussed in Section 6.4. Finally, in Section 6.5 we present a general application of Datalog in qualitative

reasoning, showing that the CSP for qualitative formalisms can be solved in polynomial time when the input has bounded treewidth.

6.1 Basics of Datalog

Fix a set of relation symbols σ. A Datalog program consists of a finite set of *rules*, traditionally written in the form

$$\phi_0 :- \phi_1, \ldots, \phi_r$$

where $\phi_0, \phi_1, \ldots, \phi_r$ are *atomic σ-formulas*, that is, formulas of the form $R(x_1, \ldots, x_n)$ for $R \in \sigma$ and variables $x_1, \ldots, x_n$. In such a rule ϕ_0 is called the *head* and $\phi_1, \ldots, \phi_r$ the *body* of the rule. The relation symbols that never appear in rule heads are called the *input relation symbols*, or *EDBs* (this term comes from database theory, and stands for *extensional database*). The other relation symbols are sometimes referred to as *IDBs*; once again, the terminology is from database theory and stands for *intensional database*.

Before we give formal definitions of the semantics of Datalog programs, we provide an instructive example, which is intended to solve $\mathrm{CSP}(\mathbb{Q}; <)$.

$$\begin{aligned} tc(x, y) &:- x < y \\ tc(x, y) &:- tc(x, u), tc(u, y) \\ \mathit{false} &:- tc(x, x) \end{aligned}$$

Here, the binary relation $<$ is the only input relation symbol, *tc* is a binary IDB (where *tc* stands for *transitive closure*), and false is a 0-ary IDB. Informally, the Datalog program computes with the help of the relation *tc* the transitive closure of the tuples in the input relation, which can be seen as a directed graph defined on the variables, with an edge from x to y if the input contains a constraint of the form $x < y$. The program derives the predicate false if and only if the input contains a directed cycle. We will give an example later on when we have discussed the semantics of Datalog.

An important measure for the complexity of a Datalog program is the maximal number k of variables per rule (Grohe, 1994). On an input structure with n elements (that is, a CSP instance with n variables), such a Datalog program can be evaluated in time $O(n^{k+1})$. This implies that a fixed Datalog program can be evaluated in time polynomial in n. A Datalog program has *width* (l, k) if all IDBs are at most l-ary, and if all rules have at most k distinct variables. The Datalog program shown above, for instance, has width $(2, 3)$. This double parameterization is less common but it is sometimes considered in the literature (Feder & Vardi, 1999). A more common parameterization is the following: a Datalog program has *width* l if it has width (l, k) for some k.

We now formally define Datalog. Our definition will be purely operational; for a semantical approach to the evaluation of Datalog programs (Kolaitis & Vardi, 1998; Ebbinghaus & Flum, 1999). Let τ be a relational signature. A Datalog program (with signature τ) is a finite set of *rules* of the form $\psi :- \phi_1, \ldots, \phi_r$, where $r \geq 0$ and where $\psi, \phi_1, \ldots, \phi_r$ are atomic τ-formulas. The formula ψ is called the *head* of the rule, and $\phi_1, \ldots, \phi_r$ is called the *body*.

An *evaluation* of a Datalog program Π on a finite structure S proceeds in steps $i = 0, 1, \ldots$; at each step i we maintain a $(\tau \cup \sigma)$-structure S^i, where σ is a finite relational signature that is disjoint from τ. The relations for the symbols from τ are always equal to the relations from S, i.e., for every $i \geq 0$ and every $R \in \tau$ we have $R^{S^i} = R^S$. For every relation symbol $R \in \sigma$ we have that $R^{S^i} \subseteq R^{S^{i+1}}$ for all $i \geq 0$. Initially, we start with the structure S^0 where all symbols from σ

denote the empty relation and all symbols in τ denote the same relation as in S. Now suppose that $R_1(u_1^1, \ldots, u_{k_1}^1), \ldots, R_l(u_1^r, \ldots, u_{k_r}^r)$ hold in S^i, and that

$$R_0(y_1^0, \ldots, y_{k_0}^0) :- R_1(y_1^1, \ldots, y_{k_1}^1), \ldots, R_l(y_1^r, \ldots, y_{k_r}^r)$$

is a rule from Π, where $u_j^i = u_{j'}^{i'}$ if $y_j^i = y_{j'}^{i'}$. Then we add the tuple $(u_1^0, \ldots, u_{k_0}^0)$ to R in S^{i+1}, where $u_j^0 = u_{j'}^i$ if and only if $y_j^0 = y_{j'}^i$. We also say that the Datalog program *derives* $R(u_1^0, \ldots, u_{k_0}^0)$ from $R_1(u_1^1, \ldots, u_{k_1}^1), \ldots, R_l(u_1^r, \ldots, u_{k_r}^r)$. The procedure stops if no new tuples can be derived. Note that a Datalog program Π always terminates: if Π has width (l, k) then we know that it can be evaluated in $O(n^{k+1})$ time. The termination property also follows from the fact that we can derive at most n^l tuples per relation.

Let us now apply the Datalog program for CSP($\mathbb{Q}; <$) to a concrete example. Consider the input instance

$$\exists v_1, v_2, v_3, v_4 (v_1 < v_2 \wedge v_2 < v_3 \wedge v_3 < v_4 \wedge v_4 < v_1)$$

When applying the program to this instance, we may first assume that tc is initiated such that it includes the tuples $(v_1, v_2), (v_2, v_3), (v_3, v_4), (v_4, v_1)$. This is accomplished by applying the rule $tc(x, y) :- x < y$ to the input a number of times. After this, the rule $tc(x, y) :- tc(x, u), tc(u, y)$ is applicable: we can for instance add the tuple (v_1, v_3) to tc since $(v_1, v_2) \in tc$ and $(v_1, v_3) \in tc$. By repeating this a number of times, the tuple (v_1, v_1) will be added to tc and this makes the rule $\mathit{false} :- tc(x, x)$ applicable. When the computation stops (that is, when no new tuples can be derived), we can check whether false has been derived or not. This is indeed the case in this example and we conclude that the instance has no solution. The rules in a Datalog program are applied non-deterministically so the computation above can be performed in many different orders. It is important to observe that the actual order does not matter: whenever false can be derived, the given instance has no solution.

6.2 A Datalog Program for Path Consistency

We will now give an important example of a Datalog program. One of the main algorithmic techniques used in temporal and spatial reasoning is consistency checking. Many different consistency procedures can be formulated with Datalog programs and we illustrate this for path consistency.

Let $\mathbf{A} = (A; \vee, -, 0, 1, \circ, ^\smile, \mathrm{Id})$ be a finite abstract relation algebra as defined in Definition 7. Recall that the empty set is denoted 0 in this context and that we implicitly have access to the intersection operator $\wedge$. The path-consistency procedure (see Figure 7) for $\mathbf{A}$ takes as input an $\mathbf{A}$-network N, enforces path-consistency, and returns false if the relation between two variables is empty, and otherwise returns the modified $\mathbf{A}$-network. The execution of the procedure on N only depends on $\mathbf{A}$ as an *abstract* relation algebra (and not on particular representations of $\mathbf{A}$).

It is quite straightforward to reformulate the path-consistency procedure as a Datalog program Π. First of all, note that the loop condition "Loop until no further changes" matches the interruption criterion for Datalog programs (i.e., no more tuples can be derived). Secondly, the exact choices of nodes x, y, z is not specified and this matches the nondeterministic nature of Datalog programs. Hence, we let the EDBs of Π consist of a binary relation symbol R_a for each element $a \in A$, and as IDBs, we have a binary relation S_a for each $a \in A$ together with the distinguished 0-ary predicate false.

The path-consistency program Π contains the rule

$$S_a(x, y) :- R_a(x, y)$$

$\mathrm{PC}_{\mathbf{A}}(N)$
Input: an $\mathbf{A}$-network $N = (V, f)$.
Do
  For all distinct nodes $x, y, z \in V$:
    Replace $f(x, y)$ by $f(x, y) \wedge (f(x, z) \circ f(z, y))$
    If $f(x, y) = 0$ then **reject**
Loop until no further changes
Return (V, f).

Figure 7: The path-consistency procedure for a relation algebra $\mathbf{A}$.

for each $a \in A$, that is, we "copy" the input into the "internal" relations that are used for the computation. Next, we have the rules

$$S_{a \circ b}(x, y) :- S_a(x, z), S_b(z, y)$$
$$S_{a \wedge b}(x, y) :- S_a(x, y), S_b(x, y)$$

for all $a, b \in A$. The first rule says that if the relation a holds between x and z and the relation b holds between z and y, then we can infer that the relation $a \circ b$ holds between x and y. From the Datalog viewpoint, this piece of information is represented by extending the EDB $S_{a \circ b}$ with the tuple (x, y). The second rule works analogously. Finally, we have the rule

$$\mathit{false} :- S_0(x, y) \ .$$

that captures the rejection criterion: if the relation between two variables is empty, then there is no solution. Given an $\mathbf{A}$-network N, the program Π derives *false* if and only if the path-consistency procedure rejects when applied to N. Clearly, this construction can easily be generalised to k-consistency for arbitrary $k \geq 1$.

A program of this kind can also be used for "simplifying" CSPs in a way that is similar to ordinary consistency-enforcing procedures: if the program cannot prove that the given instance is not solvable, then one can extract the information that has been inferred from the instance. We know that the relation S_a contains all tuples (x, y) such that the program has inferred that x and y are related by a. It follows that the inferred relation between x and y is $\bigwedge\{a \in A \mid (x, y) \in S_a\}$.

In Section 5 we have seen *weak composition* as an approximation of ordinary composition. Clearly, path consistency can be based on weak composition instead of ordinary composition; this is sometimes referred to as the *algebraic closure* in the literature. The modification of the algorithm in Figure 7 is straightforward: let $\diamond$ denote the weak composition operator and change the line "Replace $f(x, y)$ by $f(x, y) \wedge (f(x, z) \circ f(z, y))$" into "Replace $f(x, y)$ by $f(x, y) \wedge (f(x, z) \diamond f(z, y))$". The Datalog program can be modified analogously by changing the rule $S_{a \circ b}(x, y) :- S_a(x, z), S_b(z, y)$ in Π into $S_{a \diamond b}(x, y) :- S_a(x, z), S_b(z, y)$.

6.3 Canonical Datalog Programs

Canonical Datalog programs was introduced by Feder and Vardi (1999) for constraints over finite domains and the idea was generalised to ω-categorical structures by Bodirsky and Dalmau (2013).

It is interesting to note that the construction uses ω-categoricity in an essential way: Theorem 19 is needed in the construction.

For $l, k \in \mathbb{N}$, the *canonical* (l,k)*-Datalog program* for an ω-categorical structure Γ with finite relational signature contains an IDB for every at most l-ary primitive positive definable relation in Γ – by Theorem 19, there are only finitely many of those. The empty 0-ary relation, primitively positively defined by $\bot$, serves as $false$. The input relation symbols are precisely the symbols from τ.

Let Γ' be the structure obtained from Γ by adding all at most l-ary primitive positive definable relations to Γ. Such first-order expansions of ω-categorical structures are again ω-categorical by Theorem 19(4) since Γ and Γ' have the same automorphisms by Proposition 16. The new relations of Γ' will be the IDBs and the relations that were already present in Γ are the EDBs of the canonical Datalog program. Let us compare this with the path-consistency procedure: Γ contains the relations that are allowed in the input instance while Γ' contains all relations that may appear while running the path-consistency procedure. Theorem 19 implies that over Γ' there is a finite number of inequivalent formulas $\Psi(\overline{x})$ of the form

$$\big(\exists \overline{y}(\psi_1(\overline{x},\overline{y}) \wedge \cdots \wedge \psi_j(\overline{x},\overline{y}))\big) \rightarrow R(\overline{x})$$

having at most k distinct variables and where $\psi_1, \ldots, \psi_j$ denote atomic formulas of the form $R_1(\overline{z}_1), \ldots, R_j(\overline{z}_j)$ for IDBs or EDBs $R_1, \ldots, R_j$ and an IDB R. For each of these inequivalent implications $\Psi(\overline{x})$ we introduce a rule

$$R(\overline{x}) :- R_1(\overline{z}_1), \ldots, R_j(\overline{z}_j)$$

into the canonical Datalog program if $\forall \overline{x}.\Psi(\overline{x})$ is valid in Γ'. In other words, we introduce this rule if $R(\overline{x})$ is implied by $\exists \overline{y}\big(\psi_1(\overline{x},\overline{y}) \wedge \cdots \wedge \psi_j(\overline{x},\overline{y})\big)$ in Γ'. Since there are finitely many implications Ψ that are pairwise inequivalent in Γ', the canonical (l,k)-Datalog program is finite. Let us once again compare with the path-consistency procedure: these implications correspond to the information inferred by computing intersections and compositions.

The final stage of the evaluation of the canonical Datalog program Π on a given instance S of $\mathrm{CSP}(\Gamma)$ gives rise to an instance S' of $\mathrm{CSP}(\Gamma')$. Since the IDBs of Π are relations from Γ', S' is an instance of $\mathrm{CSP}(\Gamma')$ and S' is satisfiable if and only if S is satisfiable. Consequently, we can view canonical Datalog programs as a kind of consistency-enforcing procedures. The applicability of the canonical (l,k)-Datalog program is captured in the following result.

Theorem 36 (Bodirsky & Dalmau, 2013)**.** *Let Γ be a finite or countably infinite ω-categorical structure with finite relational signature. Then $\mathrm{CSP}(\Gamma)$ can be solved by an (l,k)-Datalog program if and only if $\mathrm{CSP}(\Gamma)$ can be solved by the canonical (l,k)-Datalog program.*

Hence, from a theoretical point of view it is sufficient to study canonical Datalog programs. From a practical and implementational point of view, this is not in general true. We have a situation very similar to that of arc- and path-consistency: the basic ideas are simple but a lot of work and ingenuity are needed for transforming these ideas into practical solutions suitable for solving scientific and industrial problems.

Finally, we would like to point out the close connection between orbital relation algebras (as defined in Section 5) and the canonical $(2,3)$ program. Let Γ be an ω-categorical binary structure which is a model-complete core (see Sec. 3.3), and let $\mathbf{A}$ be the orbital relation algebra for Γ. Then the Datalog program for the path consistency algorithm for $\mathbf{A}$ equals the canonical $(2,3)$-Datalog program for Γ, since the orbitals are primitive positive definable in Γ (Theorem 21).

6.4 Datalog and Primitive Positive Interpretations

We have seen (in Lemmas 23 and 24) that there are close connection between ω-categoricity, computational complexity and interpretations, i.e., we have

- if a structure Δ is ω-categorical, then every structure Γ that is first-order interpretable in Δ is ω-categorical and
- there is a polynomial-time reduction from CSP(Γ) to CSP(Δ) whenever Γ has a primitive positive interpretation in Δ.

Primitive positive definability is additionally an important tool for identifying CSPs that can be solved by Datalog. The following result is well-known for CSPs with finite domains; however, we have not been able to find an explicit reference in the literature. The statement also holds for CSPs over infinite domains so, for the convenience of the reader, we present a proof for both finite and infinite domains. Note that canonical Datalog programs is an essential ingredient in the proof.

Proposition 37. *Let Γ be an arbitrary (finite or infinite) relational structure such that $\mathrm{CSP}(\Gamma)$ can be solved by Datalog. Suppose that there is a primitive positive interpretation of a structure Δ in Γ. Then $\mathrm{CSP}(\Delta)$ can be solved by Datalog, too.*

Proof. Since $\mathrm{CSP}(\Gamma)$ can be solved by Datalog, by Theorem 36 there exist $l, k \in \mathbb{N}$, $l \leq k$, such that the canonical (l,k)-Datalog program Π solves $\mathrm{CSP}(\Gamma)$. Let d be the dimension of the primitive positive interpretation I of the τ-structure Δ in the σ-structure Γ, let $\delta_I(x_1, \dots, x_d)$ be the domain formula, and let $h\colon \delta_I(\Gamma^d) \to D(\Delta)$ be the coordinate map. Consider the expansion Δ' of Δ that contains for every primitive positive formula $\phi_I(x_{1,1}, \dots, x_{d,k})$ in Γ the k-ary relation defined by ϕ_I on $\delta_I(\Gamma^d)$. We show that the canonical (l,k)-Datalog program Π' for Δ' solves $\mathrm{CSP}(\Delta)$.

Let ϕ be an instance of $\mathrm{CSP}(\Delta)$ with variable set $U = \{x_1, \dots, x_n\}$. If ϕ is satisfiable in Δ, then it is satisfiable in the expansion Δ', too, and therefore the canonical Datalog program for Δ' accepts ϕ (Theorem 36). So let us assume that ϕ is unsatisfiable in Δ. We begin by defining an instance ψ of $\mathrm{CSP}(\Gamma)$. This instance will be used as a "guide" when we inductively show that Π' applied to ϕ derives false.

For fresh and pairwise distinct variables $V := \{y_j^i \mid 1 \leq i \leq d \text{ and } 1 \leq j \leq n\}$ we set ψ_1 to be the formula

$$\bigwedge_{1 \leq i \leq n} \delta_I(y_i^1, \dots, y_i^d) \,.$$

Let ψ_2 be the conjunction of the formulas $\theta_I(y_{i_1}^1, \dots, y_{i_1}^d, \dots, y_{i_k}^1, \dots, y_{i_k}^d)$ over all conjuncts $\theta = R(x_{i_1}, \dots, x_{i_k})$ of ϕ. By moving existential quantifiers to the front, the sentence

$$\exists y_1^1, \dots, y_n^d \, (\psi_1 \wedge \psi_2)$$

can be re-written to a primitive positive σ-formula ψ.

We claim that ψ is unsatisfiable in Γ. Let C be the domain of Γ, B the domain of Δ, and suppose for contradiction that $f\colon V \to C$ satisfies all conjuncts of ψ in Γ. Hence, by construction of ψ, if ϕ has a conjunct $\theta = R(x_{i_1}, \dots, x_{i_k})$, then

$$\Gamma \models \theta_I((f(y_{i_1}^1), \dots, f(y_{i_1}^d)), \dots, (f(y_{i_k}^1), \dots, f(y_{i_k}^d))) \,.$$

By the definition of interpretations, this implies that

$$\Delta \models R(h(f(y^1_{i_1}), \dots, f(y^d_{i_1})), \dots, h(f(y^1_{i_k}), \dots, f(y^d_{i_k}))) \; .$$

Hence, the mapping $g\colon U \to B$ that sends x_i to $h(f(y^1_i), \dots, f(y^d_i))$ satisfies all conjuncts of ϕ in Δ, in contradiction to the assumption that ϕ is satisfiable.

Since Π solves $\mathrm{CSP}(\Gamma)$ we consequently have that Π applied to ψ derives *false*. We use this derivation to show that Π' applied to ϕ derives *false*, too. Suppose that Π derives a fact $R(\bar{a})$, where $\bar{a}$ is a tuple of variables from ψ of length at most l; to avoid proliferation of symbols in the proof, we assume that the arity of R is exactly l. By the definition of canonical Datalog programs, R has been introduced in Π for a primitive positive formula μ in the language of Γ. By definition of ψ, for each $i \leq l$ there exists $j_i \leq n$ such that a_i appears among the variables $y^1_{j_i}, \dots, y^d_{j_i}$ of ψ. By definition of Δ' there exists a relation R' in the signature of Δ' that denotes the relation defined by the primitive positive formula $\mu \wedge \bigwedge_{i,j} y^i_{j_i} = y^i_{j_i}$ (the purpose of the last part of the formula is to make sure that certain variables appear in the formula so that the defined relation has arity ld). Moreover, by the definition of canonical Datalog programs, Π' has an IDB that has been introduced for this relation, which we also denote by R'. We claim that Π' derives $R'(x_{j_1}, \dots, x_{j_l})$. We show this by induction over the evaluation of Π on ψ.

Suppose that $R(\overline{a})$ has been derived by the rule

$$R(\overline{z}_0) :- R_1(\overline{z}_1), \dots, R_s(\overline{z}_s)$$

in the evaluation of Π on ψ. It suffices to distinguish the cases that all of $R_1, \dots, R_s$ are EDBs (the base case of the induction), and the case that all of $R_1, \dots, R_s$ are IDBs (the induction step). In the first case, each R_i of arity k can be defined by a primitive positive formula $\phi_I(x_1, \dots, dk)$ over Γ.

In the second case, for the sake of notation, assume that the relation symbols $R_1, \dots, R_s$ all have arity l. Let $\bar{b}_1, \dots, \bar{b}_s$ be l-tuples of variables from ψ that are the witnesses for $\overline{z}_1, \dots, \overline{z}_s$ showing that the rule was applicable. By the definition of ψ, for each $r \leq s$ and $i \leq l$ there exists $j_{i,r} \leq n$ such that $b_{i,r}$ appears among the variables $y^1_{j_{i,r}}, \dots, y^d_{j_{i,r}}$ of ψ. By the inductive hypothesis, we know that Π' already derived $R'_r(x_{j_{1,r}}, \dots, x_{j_{l,r}})$ for all $r \leq s$. We claim that

$$R'_i(\overline{z}_0) :- R'_1(\overline{z}_1), \dots, R'_s(\overline{z}_s) \tag{2}$$

is a rule in Π'. For $r \in \{1, \dots, s\}$, let η_r be the primitive positive formula with ld free variables that defines the relation R'_r over Γ. Let δ be the domain formula δ_I of the interpretation, but without the existential quantifiers. Now the claim follows from the fact that $\bigwedge_{r \leq s}(\eta_r(\overline{z}_r) \wedge \delta(\overline{z}_r))$ implies $\theta(\overline{z}_0)$ in Γ. Using Rule (2) the program Π' infers $R'(x_{j_1}, \dots, x_{j_l})$ from the facts $R'_1(x_{j_{1,1}}, \dots, x_{j_{l,1}}), \dots,$ $R'_s(x_{j_{1,s}}, \dots, x_{j_{l,s}})$.

Note that when R is the distinguished 0-ary predicate *false* of Π, then the formula μ defines the empty relation in Γ, and hence the relation R' denotes the empty 0-ary relation in Δ', and the corresponding IDB R' is the distinguished goal predicate of Π'. Hence, Π' derives *false*, too, which is what we had to show. □

Note that the proof of Proposition 37 also shows that primitive positive interpretations preserve the Datalog width of the corresponding CSPs. One may use Proposition 37 for obtaining quite useful results. The following is one example.

Corollary 38. *Let Γ be any structure with finite relational signature, and let Δ be the dual encoding of Γ. Then $\mathrm{CSP}(\Gamma)$ can be solved by Datalog if and only if $\mathrm{CSP}(\Delta)$ can be solved by Datalog.*

Proof. We have already mentioned in Section 3.4 that there is a primitive positive interpretation of Γ in Δ and vice versa. The statement now follows from Proposition 37. □

6.5 Structurally Restricted CSPs

We would like to mention an interesting application of Datalog for qualitative CSPs. It is well-known that many hard computational problems become polynomial-time solvable when the input is restricted to structures of bounded treewidth (for a formal definition of the tree-width of a structure, we refer to the article by Bodirsky & Dalmau, 2013). This is in particular true for the CSPs of finite structures. Bodirsky and Dalmau (2013) have shown that this fact extends to all ω-categorical infinite-domain CSPs. This is a powerful result given the large number of relevant CSPs that can be formulated with ω-categorical structures. The result generalises work by Dalmau, Kolaitis, and Vardi (2002) for the situation where Γ is finite, which in turn builds on work by Freuder (1990).

Theorem 39. *Let Γ be an ω-categorical structure. Then every instance A of $\mathrm{CSP}(\Gamma)$ whose core has tree-width at most l can be solved in polynomial-time with a Datalog program of width l.*

Note that it makes sense to speak of the "core of an instance A" since A can be viewed as a relational structure, just as in Section 2.1.

7. Renz's Challenge

We now come back to Renz's challenge that was presented and reformulated in the introduction: for which ω-categorical structures Γ is Datalog a sound and complete decision procedure for $\mathrm{CSP}(\Gamma)$? In the case of finite-domain CSPs, this question has been answered completely. Thus, we first revisit the situation in the finite and continue by discussing potential generalisations to qualitative infinite-domain CSPs.

7.1 Datalog for Finite Templates

Feder and Vardi (1999) gave examples of finite structures Γ whose CSP cannot be solved by Datalog.

Theorem 40. *Let $(A, +, 0)$ be a finite Abelian group and $a \in A \setminus \{0\}$. Let S_A be the structure $(A; \{(x, y, z) \mid x + y = z + a\}, \{0\})$. Then $\mathrm{CSP}(S_A)$ cannot be solved by Datalog.*

Proof. The structure S_A is an example of a structure with the *ability to count* and the result follows from Feder and Vardi (1999). The result also follows from more powerful non-expressibility results of Atserias, Bulatov, and Dawar (2009). □

Corollary 41. *Let Γ be an arbitrary (finite or infinite) relational structure, and suppose that Γ interprets primitively positively a structure that is homomorphically equivalent to S_A for a finite abelian group A. Then $\mathrm{CSP}(\Gamma)$ cannot be solved by Datalog.*

Proof. Let Δ be the structure that is homomorphically equivalent to S_A and that has a primitive positive interpretation in Γ. Since homomorphic equivalence does not change the CSP, Theorem 40 implies that $\mathrm{CSP}(\Delta)$ cannot be solved by Datalog. Proposition 37 then implies that $\mathrm{CSP}(\Gamma)$ cannot be solved by Datalog, either. □

Let Γ be a structure with a finite domain and a finite relational signature. Larose and Zádori (2007) conjectured in 2007 (in a slightly different but equivalent form; a discussion of the various formulations has been given recently by Barto, Opršal, & Pinsker, 2015) that $\mathrm{CSP}(\Gamma)$ can be solved by Datalog if and only if Γ does not satisfy the condition given in Corollary 41. The conjecture has been proved in the affirmative by Barto and Kozik (2014). There are also polymorphism characterisations of those CSPs over a finite domain D that can be solved by Datalog. For this, we need the following notion: a function $f\colon D^n \to D$, for $n \geq 2$, is called a *weak near unanimity* if for all $x, y \in D$ we have

$$f(x, \ldots, x, y) = f(x, \ldots, x, y, x) = \cdots = f(y, x, \ldots, x).$$

Note that we do not require that $f(x, \ldots, x) = x$.

Theorem 42 (Barto & Kozik, 2014; Kozik, Krokhin, Valeriote, & Willard, 2015)**.** *Let Δ be a structure with a finite domain and a finite relational signature. Then the following are equivalent.*

- $\mathrm{CSP}(\Delta)$ *can be solved by Datalog;*
- *there exists an $n_0 \geq 2$ such that Δ has for all $n \geq n_0$ a weak near unanimity polymorphism of arity n;*
- *Δ has weak near polymorphisms f and g of arity 3 and 4 such that for all $x, y \in D$*

$$f(x, x, y) = g(x, x, x, y)\ .$$

This result strengthens many isolated results about applicability of Datalog that have been known before. For example, the result implies that finite structures with a *semilattice polymorphism* or a *near unanimity* polymorphism have CSPs that are solvable by Datalog. Here, a binary polymorphism f is a semilattice polymorphism if it satisfies $f(x, x) = x$, $f(x, f(y, z)) = f(f(x, y), z)$, and $f(x, y) = f(y, x)$, and it is a near unanimity polymorphism if it is a weak near unanimity polymorphisms that additionally satisfies $f(y, x, \ldots, x) = x$.

Since the universal-algebraic approach in principle also applies to ω-categorical templates (as discussed in Section 3.5), one might be optimistic about the chances of proving a Barto-Kozik result for infinite-domain qualitative CSPs, too.

7.2 Datalog for Qualitative Formalisms

Some of the results about solvability of CSPs by Datalog generalise from finite to qualitative infinite-domain CSPs. For example, we have the following.

Theorem 43 (Bodirsky & Dalmau, 2013)**.** *Let Γ be an ω-categorical structure that has a weak near-unanimity polymorphism f that additionally satisfies $f(y, x, \ldots, x) = f(x, \ldots, x)$. Then $\mathrm{CSP}(\Gamma)$ can be solved by Datalog.*

Unfortunately, it is not clear how to generalise the result of Barto and Kozik to qualitative infinite-domain CSPs. Indeed, the following shows that the situation for qualitative formalisms differs fundamentally from the situation for finite-domain CSPs.

Proposition 44. *There is a binary ω-categorical CSP that has a semilattice polymorphism but that cannot be solved by Datalog.*

Proof. The structure $\Gamma := (\mathbb{Q}; \{(x, y, z) \mid x > y \vee x > z\})$ is ω-categorical by Lemma 23, and it has been shown (Bodirsky & Kára, 2010) that $\mathrm{CSP}(\Gamma)$ cannot be solved by Datalog. It is easy to verify that the maximum function $(x, y) \mapsto \max(x, y)$ is a polymorphism of Γ, and that this function is a semilattice operation. In order to obtain a *binary* ω-categorical structure with this property, recall that the dual encoding Δ of Γ is primitive positive interpretable in Γ, and vice versa. Again Δ is ω-categorical by Lemma 23, and $\mathrm{CSP}(\Delta)$ cannot be solved by Datalog by Proposition 37. It is straightforward to verify that $\max$ applied componentwise to the elements of Δ is a semilattice polymorphism of Δ. □

In contrast, as we have stated before, the existence of a semilattice polymorphism implies solvability by Datalog (Jeavons et al., 1997) if we restrict ourselves to finite domains.

Finally, we mention that another universal-algebraic condition has been discovered recently that implies that a CSP of an ω-categorical structure Γ can be solved by Datalog (Bodirsky & Mottet, 2016). The condition given there allows to reduce $\mathrm{CSP}(\Gamma)$ to $\mathrm{CSP}(\Delta)$ for a finite structure Δ that satisfies the conditions of Theorem 42. Since $\mathrm{CSP}(\Delta)$ can then be solved by Datalog, and since the reduction preserves solvability by Datalog, $\mathrm{CSP}(\Gamma)$ can be solved by Datalog, too. This implies that the polymorphism condition from Theorem 42 can be translated into a powerful polymorphism condition for Γ that implies solvability by Datalog; we refer the reader to Theorem 8 in the paper by Bodirsky and Mottet (2016) and the following comment.

8. Conclusions and Research Directions

The model-theoretic concept of ω-categoricity underlies both the proposed definition of qualitative CSPs and the ability to use Datalog for solving qualitative CSPs. The question "which qualitative reasoning problems can be solved by local consistency methods" that we discussed (and reformulated) in Section 1 is now the mathematically precise question: "for which ω-categorical structures Γ can $\mathrm{CSP}(\Gamma)$ be solved by Datalog?". To answer this question we have encountered a number of mathematical tools. However, a series of problems remains open before we can answer this question. Let us highlight the following.

Problem 1. *Find conditions on ω-categorical structures that include those having weak near unanimity polymorphisms satisfying $f(y, x, \ldots, x) = f(x, \ldots, x)$ (refer to Theorem 43) and those satisfying the assumptions of Theorem 8 in the paper by Bodirsky and Mottet (2016), but that do not include all structures having a semilattice polymorphism (see Proposition 44).*

We now present a list of concrete classification projects. We believe that these projects are feasible and that solutions to them will lead to substantial insight into phenomena that are relevant for all qualitative CSPs.

Problem 2. *Let Γ be a relational structure with the same domain $\mathbb{I}$ as the standard representation of Allen's Interval Algebra (Section 2.5), and such that all relations in Γ are first-order definable over the relations of Allen's Interval Algebra. When is $\mathrm{CSP}(\Gamma)$ solvable by Datalog? When is it in P?*

An even more ambitious problem is the following.

Problem 3. *Classify the computational complexity of $\mathrm{CSP}(\Gamma)$ when Γ has a first-order interpretation in $(\mathbb{Q}; <)$. When is $\mathrm{CSP}(\Gamma)$ solvable by Datalog? When is it in P?*

Since Allen's Interval Algebra has a first-order interpretation in $(\mathbb{Q}; <)$, a solution to Problem 3 would also imply a solution to Problem 2. An important step towards solving Problem 2 is to prove the following conjecture.

Conjecture 1. *Let Γ be as in Problem 2, let Δ be the model-complete core of Γ, and let M be the endomorphism monoid of Δ. Then there are only finitely many monoids M that can arise in this way.*

Results analogous to this conjecture have provided the cornerstone of several complexity classification projects such as the full classifications of temporal constraints (Bodirsky & Kára, 2009), Graph-SAT problems (Bodirsky & Pinsker, 2015a), and phylogeny constraints (Bodirsky, Jonsson, and Pham, 2016). We can, naturally, suggest a similar project for RCC-5.

Problem 4. *Let Γ be a relational structure with the same domain as the ω-categorical representation RCC-5$_{\omega\text{-cat}}$ of RCC-5 (Section 5.3), and such that all relations in Γ are first-order definable over RCC-5$_{\omega\text{-cat}}$. When is $\mathrm{CSP}(\Gamma)$ solvable by Datalog? When is it in P?*

One may object that one should use a more powerful formalism (such as RCC-8) as the basis for such a project. However, our understanding of RCC-8 is quite weak — in fact, we do not even know the tractable subclasses of RCC-8. Thus, it seems like a better idea to start studying a simpler formalism. Once again, it will be crucial to understand the model-complete cores of the structures Γ.

Conjecture 2. *Let Γ be as in Problem 4, let Δ be the model-complete core of Γ, and let M be the endomorphism monoid of Δ. Then there are only finitely many monoids M that can arise in this way.*

Our final question concerns the definition of qualitative CSPs that we proposed in Section 4.4: is it a reasonable way of capturing the underlying intuitive concept from Section 4?

Acknowledgements

The authors would like to thank Johannes Greiner and Matthias Westphal for valuable feedback on a draft of this article and the reviewers for their detailed comments. The first author has received funding from the European Research Council (Grant Agreement number 681988, CSP-Infinity) and from the German Science Foundation (DFG, project number 622397).

References

Abiteboul, S., Hull, R., & Vianu, V. (1995). *Foundations of Databases*. Addison-Wesley.

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

Amaneddine, N., Condotta, J., & Sioutis, M. (2013). Efficient approach to solve the minimal labeling problem of temporal and spatial qualitative constraints. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI-2013)*, 696–702.

Apt, K. R., & Brand, S. (2006). Infinite qualitative simulations by means of constraint programming. In *Proc. 12th International Conference on Principles and Practice of Constraint Programming (CP-2006)*, pp. 29–43.

Atserias, A., Bulatov, A. A., & Dawar, A. (2009). Affine systems of equations and counting infinitary logic. *Theoretical Computer Science*, *410*(18), 1666–1683.

Balbiani, P., Condotta, J., & del Cerro, L. F. (1998). A model for reasoning about bidemsional temporal relations. In *Proc. 6th International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, pp. 124–130.

Balbiani, P., Condotta, J., & del Cerro, L. F. (1999). A tractable subclass of the block algebra: Constraint propagation and preconvex relations. In *Proc. 9th Portuguese Conference on Artificial Intelligence (EPIA-99)*, pp. 75–89.

Barto, L., & Kozik, M. (2014). Constraint satisfaction problems solvable by local consistency methods. *Journal of the ACM*, *61*(1), 3:1–3:19.

Barto, L., Opršal, J., & Pinsker, M. (2015). The wonderland of reflections. Preprint arXiv:1510.04521.

Bienvenu, M., ten Cate, B., Lutz, C., & Wolter, F. (2014). Ontology-based data access: A study through disjunctive Datalog, CSP, and MMSNP. *ACM Transactions on Database Systems*, *39*(4), 33:1–33:44.

Bodirsky, M. (2007). Cores of countably categorical structures. *Logical Methods in Computer Science*, *3*(1), 1–16.

Bodirsky, M. (2008). Constraint satisfaction problems with infinite templates. In Vollmer, H. (Ed.), *Complexity of Constraints (a collection of survey articles)*, Vol. 5250 of *Lecture Notes in Computer Science*, pp. 196–228. Springer.

Bodirsky, M. (2012). Complexity classification in infinite-domain constraint satisfaction. Mémoire d'habilitation à diriger des recherches, Université Diderot – Paris 7. Available at arXiv:1201.0856.

Bodirsky, M., & Chen, H. (2007). Qualitative temporal and spatial reasoning revisited. In *Proc. 21st International Workshop on Computer Science Logic (CSL-2007)*, pp. 194–207.

Bodirsky, M., & Chen, H. (2009). Qualitative temporal and spatial reasoning revisited. *Journal of Logic and Computation*, *19*(6), 1359–1383.

Bodirsky, M., & Dalmau, V. (2013). Datalog and constraint satisfaction with infinite templates. *Journal on Computer and System Sciences*, *79*, 79–100.

Bodirsky, M., & Grohe, M. (2008). Non-dichotomies in constraint satisfaction complexity. In *Proc. 35th International Colloquium on Automata, Languages and Programming (ICALP-2008)*, pp. 184 –196.

Bodirsky, M., & Hils, M. (2012). Tractable set constraints. *Journal of Artificial Intelligence Research*, *45*, 731–759.

Bodirsky, M., Hils, M., & Martin, B. (2010). On the scope of the universal-algebraic approach to constraint satisfaction. In *Proc. 25th Annual IEEE Symposium on Logic in Computer Science (LICS-2010)*, pp. 90–99. IEEE Computer Society.

Bodirsky, M., Jonsson, P., & Pham, V. T. (2016). The complexity of phylogeny constraint satisfaction. In *Proc. 33rd Symposium on Theoretical Aspects of Computer Science, STACS-2016*, pp. 20:1–20:13.

Bodirsky, M., & Kára, J. (2009). The complexity of temporal constraint satisfaction problems. *Journal of the ACM*, *57*(2), 1–41.

Bodirsky, M., & Kára, J. (2010). A fast algorithm and Datalog inexpressibility for temporal reasoning. *ACM Transactions on Computational Logic*, *11*(3), 15:1-15:21.

Bodirsky, M., & Mottet, A. (2016). Reducts of finitely bounded homogeneous structures, and lifting tractability from finite-domain constraint satisfaction. In *Proc. 31st Annual Symposium on Logic in Computer Science (LICS-2016)*, pp. 623–632.

Bodirsky, M., & Mueller, J. K. (2011). Rooted phylogeny problems. *Logical Methods in Computer Science*, *7*(4).

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

Bodirsky, M., & Pinsker, M. (2015a). Schaefer's theorem for graphs. *Journal of the ACM*, *62*(3), 52 pages (article number 19).

Bodirsky, M., & Pinsker, M. (2015b). Topological Birkhoff. *Transactions of the American Mathematical Society*, *367*, 2527–2549.

Bodirsky, M., & Woelfl, S. (2011). RCC8 is tractable on instances of bounded treewidth. In *Proc. 22nd International Joint Conferences on Artificial Intelligence (IJCAI-2011)*, pp. 756–761.

Bulatov, A. A. (2006). A dichotomy theorem for constraint satisfaction problems on a 3-element set. *Journal of the ACM*, *53*(1), 66–120.

Bulatov, A. A., Krokhin, A. A., & Jeavons, P. G. (2005). Classifying the complexity of constraints using finite algebras. *SIAM Journal on Computing*, *34*, 720–742.

Cameron, P. J. (1990). *Oligomorphic permutation groups*. Cambridge University Press, Cambridge.

Cantor, G. (1884). Über unendliche, lineare Punktmannigfaltigkeiten. *Mathematische Annalen*, *23*, 453–488.

Ceri, S., Gottlob, G., & Tanca, L. (1989). What you always wanted to know about datalog (and never dared to ask). *IEEE Transactions on Knowledge and Data Engineering*, *1*(1), 146–166.

Cohen, D., Jeavons, P., Jonsson, P., & Koubarakis, M. (2000). Building tractable disjunctive constraints. *Journal of the ACM*, *47*(5), 826–853.

Cohen, D. A., Cooper, M. C., Jeavons, P. G., & Zivny, S. (2015). Binarisation via dualisation for valued constraints. In *Proc. 29th AAAI Conference on Artificial Intelligence (AAAI-2015)*, pp. 3731–3737.

Dague, P. (1995). Qualitative reasoning: A survey of techniques and applications. *AI Communications*, *8*(3/4), 119–192.

Dalmau, V., Kolaitis, P. G., & Vardi, M. Y. (2002). Constraint satisfaction, bounded treewidth, and finite-variable logics. In *Proc. 8th International Conference on Principles and Practice of Constraint Programming (CP-2002)*, pp. 310–326.

Dechter, R. (2003). *Constraint Processing*. Morgan Kaufmann.

Drakengren, T., & Jonsson, P. (1998). Reasoning about set constraints applied to tractable inference in intuitionistic logic. *Journal of Logic and Computation*, *8*(6), 855–875.

Düntsch, I. (2005). Relation algebras and their application in temporal and spatial reasoning. *Artificial Intelligence Review*, *23*, 315–357.

Düntsch, I., Wang, H., & McCloskey, S. (1999). Relations algebras in qualitative spatial reasoning. *Fundamenta Informaticae*, *39*(3), 229–248.

Düntsch, I., Wang, H., & McCloskey, S. (2001). A relation algebraic approach to the region connection calculus. *Theoretical Computer Science*, *255*, 63–83.

Ebbinghaus, H.-D., & Flum, J. (1999). *Finite Model Theory*. Springer, Berlin, Heidelberg, New York. 2nd edition.

Feder, T., & Vardi, M. Y. (1999). The computational structure of monotone monadic SNP and constraint satisfaction: a study through Datalog and group theory. *SIAM Journal on Computing*, *28*, 57–104.

Freuder, E. (1990). Complexity of k-tree structured constraint satisfaction problems. In *Proc. 8th National Conference on Artificial Intelligence (AAAI-90)*, pp. 4–9.

Garey, M., & Johnson, D. (1978). *A guide to NP-completeness*. CSLI Press, Stanford.

Goodman, J. E., & O'Rourke, J. (Eds.). (2004). *Handbook of Discrete and Computational Geometry* (2nd edition)., Vol. 2. Chapman & Hall/CRC.

Goyal, R. K., & Egenhofer, M. J. (2001). Similarity of cardinal directions. In *Proc. 7th International Symposium on Advances in Spatial and Temporal Databases (SSTD-2001)*, pp. 36–58.

Grohe, M. (1994). The structure of fixed-point logics. PhD-thesis at the Albert-Ludwigs Universität, Freiburg im Breisgau.

Hell, P., & Nešetřil, J. (2004). *Graphs and Homomorphisms*. Oxford University Press, Oxford.

Hirsch, R. (1996). Relation algebras of intervals. *Artificial Intelligence*, *83*, 1–29.

Hirsch, R. (1997). Expressive power and complexity in algebraic logic. *Journal of Logic and Computation*, *7*(3), 309 – 351.

Hodges, W. (1993). *Model theory*. Cambridge University Press.

Hopcroft, J. E., & Ullman, J. D. (1979). *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley.

Huang, J. (2012). Compactness and its implications for qualitative spatial and temporal reasoning. In *Proc. 13th International Conference on Principles of Knowledge Representation and Reasoning (KR-2012)*, pp. 500–508.

Huang, S. S., Green, T. J., & Loo, B. T. (2011). Datalog and emerging applications: an interactive tutorial. In *Proc. ACM SIGMOD International Conference on Management of Data (SIGMOD 2011)*, pp. 1213–1216.

Idziak, P. M., Markovic, P., McKenzie, R., Valeriote, M., & Willard, R. (2010). Tractability and learnability arising from algebras with few subpowers. *SIAM Journal on Computing*, *39*(7), 3023–3037.

Jeavons, P., Cohen, D., & Gyssens, M. (1997). Closure properties of constraints. *Journal of the ACM*, *44*(4), 527–548.

Jeavons, P. G. (1998). On the algebraic structure of combinatorial problems. *Theoretical Computer Science*, *200*, 185–204.

Jonsson, P., & Drakengren, T. (1997). A complete classification of tractability in RCC-5. *Journal of Artificial Intelligence Research*, *6*, 211–221.

Jonsson, P., & Lagerkvist, V. (2015). Upper and lower bounds on the time complexity of infinite-domain csps. In *Proc. 21st International Conference on Principles and Practice of Constraint Programming (CP-2015)*, pp. 183–199.

Kaye, R., & Macpherson, D. (Eds.). (1994). *Automorphisms of first-order structures*. Oxford University Press.

Keisler, J. (1965). Reduced products and Horn classes. *Transactions of the AMS*, *117*, 307–328.

Kolaitis, P. G., & Vardi, M. Y. (1995). On the expressive power of Datalog: Tools and a case study. *Journal of Computer and System Sciences*, *51*(1), 110–134.

Kolaitis, P. G., & Vardi, M. Y. (1998). Conjunctive-query containment and constraint satisfaction. In *Proc. 17th Symposium on Principles of Database Systems (PODS-1998)*, pp. 205–213.

Kompatscher, M., & Pham, V. T. (2016). A complexity dichotomy for poset constraint satisfaction. To appear in *Proc. 34th International Symposium on Theoretical Aspects of Computer Science (STACS-2017)*. Preprint available at *CoRR*, *abs/1603.00082*.

Koppelberg, S. (1989). Projective boolean algebras. In *Handbook of Boolean Algebras*, Vol. 3, pp. 741–773. North Holland, Amsterdam-New York-Oxford- Tokyo.

Kozik, M., Krokhin, A., Valeriote, M., & Willard, R. (2014). Characterizations of several Maltsev conditions. *Algebra Universalis*, *73*(3–4), 205–224.

Krokhin, A. A., Jeavons, P., & Jonsson, P. (2003). Reasoning about temporal relations: The tractable subalgebras of Allen's interval algebra. *Journal of the ACM*, *50*(5), 591–640.

Ladkin, P. B., & Maddux, R. D. (1994). On binary constraint problems. *Journal of the ACM*, *41*(3), 435–469.

Larose, B., & Zádori, L. (2007). Bounded width problems and algebras. *Algebra Universalis*, *56*(3-4), 439–466.

Li, S., Long, Z., Liu, W., Duckham, M., & Both, A. (2015). On redundant topological constraints. *Artificial Intelligence*, *225*, 51–76.

Ligozat, G. (1998). Reasoning about cardinal directions. *Journal of Visual Languages and Computing*, *9*(1), 23–44.

Ligozat, G. (2013). *Qualitative Spatial and Temporal Reasoning*. Wiley-ISTE.

Ligozat, G., & Renz, J. (2004). What is a qualitative calculus? A general framework. In *Proc. 8th Pacific Rim International Conference on Artificial Intelligence (PRICAI-2004)*, pp. 53–64.

Lutz, C., & Milicic, M. (2007). A tableau algorithm for description logics with concrete domains and general tboxes. *Journal of Automated Reasoning*, *38*(1-3), 227–259.

Macpherson, D. (2011). A survey of homogeneous structures. *Discrete Mathematics*, *311*(15), 1599–1634.

Marker, D. (2002). *Model Theory: An Introduction*. Springer, New York.

Marriott, K., & Odersky, M. (1996). Negative Boolean constraints. *Theoretical Computer Science*, *160*(1&2), 365–380.

Nilsson, U., & Małuszyński, J. (1990). *Logic, Programming and Prolog*. Wiley.

Ramakrishnan, R., & Ullman, J. D. (1995). A survey of deductive database systems. *Journal of Logic Programming*, *23*(2), 125–149.

Randell, D. A., Cui, Z., & Cohn, A. G. (1992). A spatial logic based on regions and connection. In *Proc. 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR-1992)*, 165–176.

Renz, J. (2012). Implicit constraints for qualitative spatial and temporal reasoning. In *Proc. 13th International Conference on Principles of Knowledge Representation and Reasoning (KR-2012)*, 509–518.

Renz, J., & Ligozat, G. (2005). Weak composition for qualitative spatial and temporal reasoning. In *Proc. 11th International Conference on Principles and Practice of Constraint Programming (CP-2005)*, pp. 534–548.

Renz, J., & Nebel, B. (2001). Efficient methods for qualitative spatial reasoning. *Journal of Artificial Intelligence Research (JAIR)*, *15*, 289–318.

Renz, J., & Nebel, B. (2007). Qualitative spatial reasoning using constraint calculi. In Aiello, M., Pratt-Hartmann, I., & van Benthem, J. (Eds.), *Handbook of Spatial Logics*, pp. 161–215. Springer Verlag, Berlin.

Renz, J., & Nebel, B. (2007). On the complexity of qualitative spatial reasoning: a maximal tractable fragment of the region connection calculus. *Artificial Intelligence*, *108*(1–2), 69–123.

Rossi, F., Petrie, C. J., & Dhar, V. (1990). On the equivalence of constraint satisfaction problems. In *Proc. 9th European Conference on Artificial Intelligence (ECAI-1990)*, pp. 550–556.

Schaefer, M. (2009). Complexity of some geometric and topological problems. In *Proc. 17th International Symposium on Graph Drawing (GD-2009)*, pp. 334–344.

Schwalb, E., & Vila, L. (1998). Temporal constraints: A survey. *Constraints*, *3*(2/3), 129–149.

Sioutis, M., & Koubarakis, M. (2012). Consistency of chordal RCC-8 networks. In *Proc. 24th International Conference on Tools with Artificial Intelligence (ICTAI-2012)*, pp. 436–443.

Tent, K., & Ziegler, M. (2012). *A course in model theory*. Lecture Notes in Logic. Cambridge University Press.

Weld, D. S., & de Kleer, J. (1990). *Readings in Qualitative Reasoning about Physical Systems*. Morgan Kaufmann, Palo Alto.

Westphal, M. (2015). *Qualitative Constraint-Based Reasoning: Methods and Applications*. Ph.D. thesis, Albert-Ludwigs-Universität Freiburg im Breisgau.

Westphal, M., Hué, J., & Wölfl, S. (2014). On the scope of qualitative constraint calculi. In *Proc. 37th Annual German Conference on AI (KI-2014)*, pp. 207–218.