University of Warsaw
Faculty of Mathematics, Informatics and Mechanics

Joanna Ochremiak

Extended constraint satisfaction problems

PhD dissertation

Supervisors

dr hab. Bartosz Klin

dr Szymon Toruńczyk

Institute of Informatics
University of Warsaw

September 2015

Author's declaration:
aware of legal responsibility I hereby declare that I have written this dissertation myself and all the contents of the dissertation have been obtained by legal means.

September 2, 2015 .....

date *Joanna Ochremiak*

Supervisors' declaration:
the dissertation is ready to be reviewed.

September 2, 2015 .....

date *dr hab. Bartosz Klin*

.....

dr Szymon Toruńczyk

Abstract

To solve an instance of the constraint satisfaction problem (CSP) one has to find an assignment of values to variables that satisfies given constraints. This thesis concerns two different extensions of the constraint satisfaction problem.

The first extension allows for an infinite number of constraints in an instance. It is phrased in the language of sets with atoms, which provides finite means of representation – an instance is founded upon a fixed infinite relational structure, and defined by finitely many first order formulas. We prove decidability for this so-called locally finite CSP, and establish tight complexity bounds in the special case when the set of possible values is finite.

We further use the constraint satisfaction framework to analyse the computational model of Turing machines with atoms (TMAs), whose alphabet, state space, and transition relation are orbit-finite sets with atoms (usually infinite but finitely presentable). We give an effective characterisation of those alphabets for which TMAs determinise, with applications to descriptive complexity.

The second extension of the CSP that we consider is known as the valued constraint satisfaction problem (VCSP). It provides a common framework for many discrete optimisation problems. We use algebraic tools to establish a necessary condition for tractability of VCSPs parametrised by sets of allowed types of constraints. We conjecture that our condition is also sufficient, and verify whether the conjecture agrees with all previously known results.

Keywords: constraint satisfaction problems, sets with atoms, valued constraint satisfaction

ACM Classification: F.2.2 [Nonnumerical Algorithms and Problems]: Computations on discrete structures; F.4.1 [Mathematical Logic]: Logic and constraint programming; F.1.1 [Models of Computation]

Streszczenie

Aby rozwiązać daną instancję problemu spełnialności więzów, należy znaleźć takie przypisanie wartości do zmiennych, żeby spełnione były wszystkie więzy. Ta rozprawa dotyczy dwóch rozszerzeń problemu spełnialności więzów.

Pierwsze rozszerzenie dopuszcza nieskończenie wiele więzów w instancji. Jest ono sformalizowane w języku teorii zbiorów z atomami, który pozwala na skończoną reprezentację – dla ustalonej nieskończonej struktury relacyjnej każda instancja zadana jest przez skończenie wiele formuł logiki pierwszego rzędu. Dowodzimy, że ten problem, który nazywamy lokalnie skończonym problemem spełnialności więzów, jest rozstrzygalny. Podajemy także ścisłe oszacowanie jego złożoności obliczeniowej w szczególnym przypadku, gdy liczba możliwych wartości jest skończona.

Następnie stosujemy narzędzia teorii spełnialności więzów do analizy modelu obliczeniowego maszyn Turinga z atomami, których alfabet, zbiór stanów, oraz relacja przejścia są orbitowo-skończonymi zbiorami z atomami (są one zazwyczaj nieskończone, ale możliwa jest ich skończona reprezentacja). Uzyskujemy efektywną charakteryzację tych alfabetów, dla których maszyny Turinga z atomami się determinizują, a ponadto pokazujemy zastosowanie tego wyniku do teorii złożoności opisowej.

Drugie rozpatrywane przez nas rozszerzenie problemu spełnialności więzów jest znane jako problem spełnialności więzów z wartościami. Pozwala ono na formalizację wielu dyskretnych problemów optymalizacyjnych. Używając narzędzi algebraicznych ustanawiamy warunek konieczny, aby problem spełnialności więzów z wartościami, sparametryzowany przez ustalony zbiór dopuszczalnych typów więzów, był rozwiązywalny w czasie wielomianowym. Formułujemy ponadto hipotezę, która głosi, że podany przez nas warunek jest jednocześnie dostateczny i sprawdzamy jej zgodność ze wszystkimi znanymi wcześniej wynikami.

Acknowledgements

First and foremost, I would like to thank my supervisors, Bartek Klin and Szymon Toruńczyk, for their constant support and encouragement, for the pleasure of working together and for everything they taught me.

I am also grateful to Mikołaj Bojańczyk who was the first to show me that theoretical computer science has beautiful mathematics.

Many thanks go to Marcin Kozik for introducing me to the world of constraint satisfaction problems, for his hospitality and the many stimulating conversations during my visits to Kraków.

I would like to thank all the people with whom I have ever worked or discussed mathematics, especially my other co-authors, Sławek Lasota, Eryk Kopczyński, Filip Mazowiecki and Adam Witkowski, for all the inspiration they gave me and for the great fun of solving problems together.

Special thanks go to my family and friends for always believing in me.

Finally, I would like to thank Jędrek, who was there for me everyday.

Dziadkowi i Tacie

Introduction

Motivation. Computer science is about effectively finding solutions that satisfy given specifications. To achieve this goal one develops algorithms that output a solution or determine its existence, analyses the complexity of those algorithms or proves that they do not exist, and looks for approximate solutions that provide the best possible trade-off between satisfying the specification and the effectiveness of an algorithm. The rich theory of computer science is developed to ultimately serve those purposes.

Every specification can be seen as a list of constraints imposed on a desired solution. The theory of constraint satisfaction provides one way of formalising this intuition. Its study begun in the 70s independently in the fields of artificial intelligence, database theory, and graph theory. And since 1978, when the seminal paper of Schaefer [45] about the complexity of boolean constraint satisfaction problems (CSPs) was published, it has become of more and more importance. The CSP framework turned out to be very robust. Its different variants appear in many research areas of theoretical computer science.

The power of constraint satisfaction lies in the fact that although it covers a large class of problems, its different variants still bear enough similarities to make transferring methods from one to another possible. Throughout the years, a very rich collection of mathematical tools got involved in the formal analysis of the CSP, from algebra, logic and model theory to probability, graph theory and combinatorics. Phrasing a problem in the constraint satisfaction framework gives one an additional insight into its structure and suggests tools that could be potentially used to solve it.

This thesis provides more evidence in favour of the unifying approach of the constraint satisfaction paradigm. It concerns two different extensions of the classical constraint satisfaction problem. Firstly, we introduce a new variant of the CSP with a possibly infinite set of constraints in an instance. We show how it arises in the analysis of a rather natural model of computation, and allows us to understand its behaviour, as well as the expressive power of an associated logic. Secondly, we successfully use universal algebraic tools developed for the decision version of the CSP to analyse its optimisation version, contributing to the understanding of its complexity.

Organisation of the thesis. Apart from this introductory chapter, which provides technical preliminaries necessary to make the subsequent development self-contained, this thesis consists of three papers:

[P1] *Locally finite constraint satisfaction problems*. Bartek Klin, Eryk Kopczyński, Joanna Ochremiak, and Szymon Toruńczyk. In Proceedings of the 30th Symposium on Logic in Computer Science (LICS'15), pages 475-486. IEEE, 2015.

In this paper we define a version of the constraint satisfaction problem with possibly infinite set of constraints. Instances are given by finitely many first order formulas and therefore can be treated as an input for algorithms. We prove a decidability result for such CSPs and provide tight complexity bounds.

[P2] *Turing machines with atoms, constraint satisfaction problems, and descriptive complexity*. Bartek Klin, Sławomir Lasota, Joanna Ochremiak, and Szymon Toruńczyk. In Proceedings of the 29th Symposium on Logic in Computer Science (LICS'14), pages 58:1-58:10. ACM, 2014.

In this paper we consider a computational model of Turing machines with atoms, which can be seen as recognisers of a special kind of relational structures. We use the framework of constraint satisfaction to establish an effective characterisation of those structures for which deterministic and nondeterministic Turing machines have the same expressive power.

[P3] *Algebraic properties of valued constraint satisfaction problem*. Marcin Kozik and Joanna Ochremiak. In Proceedings of 42nd International Colloquium on Automata, Languages, and Programming (ICALP'15), Part I, volume 9134 of LNCS, pages 846-858. Springer, 2015.

In this paper we study the complexity of the valued constraint satisfaction problem (VCSP) – an optimisation version of the CSP. We formulate a dichotomy conjecture, which says that each problem in this class is solvable in PTime or NP-hard, and proposes a criterion to distinguish those two cases. We prove the hardness direction, establishing a necessary algebraic condition for tractability of VCSPs.

In most of this thesis we work within the framework of constraint satisfaction. In the introduction below we define its basic concepts focusing on the language-restricted CSPs parametrised by a family of allowed constraints. Furthermore, we introduce sets with atoms – the key notion of papers [P1] and [P2]. Finally, in the last section we give a more detailed overview of our results.

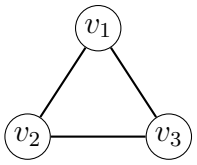
1. Constraint satisfaction problems

Many classical decision problems can be phrased as constraint satisfaction problems. An example is the problem 2-SAT, whose instances are propositional formulas such as:

$$(x \vee y) \wedge (y \vee \neg z);$$

one then asks whether a given formula has a satisfying valuation, i.e., an assignment of values 0 or 1 to variables (here, x, y and z), subject to some constraints. The constraints are given by the clauses of the formula in question. The clause $x \vee y$ says that in any valuation the pair of variables (x, y) cannot take values $(0, 0)$. The clause $y \vee \neg z$ says that the pair (y, z) cannot take values $(0, 1)$. A satisfying valuation is one that satisfies both constraints.

Another example is the problem of vertex 3-coloring, whose instances are undirected graphs such as:



Here the vertices v_1, v_2, v_3 are to be assigned colors, say, red, blue or yellow. Adjacent vertices have to be mapped to different colors. Hence, each edge in the graph imposes a constraint on the set of colorings. An edge between vertices v_i and v_j restricts possible colors of the pair (v_i, v_j) by excluding the pairs (red, red), (blue, blue), and (yellow, yellow). A valid coloring is one that satisfies all such constraints.

The following definitions provide a general framework for studying these and many other problems.

Definition 1 (CSP Instance)**.** An *instance* of the constraint satisfaction problem is a triple $(V, D, \mathcal{C})$, where V is a set of *variables*, D is a set of their possible *values*, called a *domain*, and $\mathcal{C}$ is a set of *constraints*. A *constraint* is a pair $(\bar{x}, R)$, where $\bar{x}$ is an n-tuple of variables, and R is an n-ary relation over D.

A *solution* is a mapping $f\colon V \to D$ which *satisfies* all the constraints, i.e., such that if $(\bar{x}, R) \in \mathcal{C}$ then $f(\bar{x}) \in R$, where f is applied component-wise.

Example 1. The instance of 2-SAT above is a CSP instance $(V, D, \mathcal{C})$, where: $V = \{x, y, z\}$, $D = \{0, 1\}$, $\mathcal{C} = \{((x, y), D^2 \setminus \{(0, 0)\}), ((y, z), D^2 \setminus \{(0, 1)\})\}$.

For now, let us consider only finite CSP instances, where the set of variables, the domain and the set of constraints are all finite. This problem we call the *classical* constraint satisfaction problem.

Homomorphism problem. It is well known that the constraint satisfaction problem can be seen equivalently as a homomorphism problem for relational structures. A *relational signature* Σ is a set of relational symbols. Each symbol comes with a (finite) arity. A *relational structure* $\mathbb{A}$ over Σ is a set A (called a *domain*) together with a set of relations over A, where for every symbol R in Σ of arity n, there is an n-ary relation $R^{\mathbb{A}}$. In this section we consider only finite relational structures, where both the domain and the signature are finite.

A *homomorphism* between structures $\mathbb{A}$ and $\mathbb{B}$ over the same signature is a function $f\colon A \to B$ from the domain of $\mathbb{A}$ to the domain of $\mathbb{B}$ that preserves all relations, i.e., such that if $\bar{x} \in R^{\mathbb{A}}$ then $f(\bar{x}) \in R^{\mathbb{B}}$ (where f is applied componentwise).

With every CSP instance $\mathcal{I} = (V, D, \mathcal{C})$ one can associate two relational structures $\mathbb{A}_{\mathcal{I}}$ and $\mathbb{B}_{\mathcal{I}}$. The signature $\Sigma_{\mathcal{I}}$ of both structures contains a symbol R, for every relation R that appears in any of the constraints of $\mathcal{I}$. The domain of the structure $\mathbb{A}_{\mathcal{I}}$ is the set of variables V. For each symbol R in $\Sigma_{\mathcal{I}}$ the relation $R^{\mathbb{A}}$ consists of those tuples $\bar{x}$ of variables for which $(\bar{x}, R) \in \mathcal{C}$. The structure $\mathbb{B}_{\mathcal{I}}$ has domain D and all relations R which appear in the constraints of $\mathcal{I}$. That is, $R^{\mathbb{B}} = R$, for every $R \in \Sigma_{\mathcal{I}}$. There exists a solution of $\mathcal{I}$ if and only if there is a homomorphism from $\mathbb{A}_{\mathcal{I}}$ to $\mathbb{B}_{\mathcal{I}}$, i.e., a function from V to D which maps tuples in $R^{\mathbb{A}}$ to tuples in R.

On the other hand, it is not difficult to see that every instance of the homomorphism problem can be expressed as a CSP instance.

The CSP is NP-complete in general, and the main line of research tries to identify restrictions that give rise to tractable problems (by *tractable* we usually mean polynomial-time solvable).

Let $\mathcal{A}$ and $\mathcal{B}$ be classes of relational structures. By $\mathrm{CSP}(\mathcal{A}, \mathcal{B})$ we denote the class of those CSP instances $\mathcal{I}$ where $\mathbb{A}_{\mathcal{I}} \in \mathcal{A}$ and $\mathbb{B}_{\mathcal{I}} \in \mathcal{B}$.

Example 2 (Subgraph isomorphism problem)**.** Subgraph isomorphism is a basic graph-theoretic problem: given graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, decide if G_1 is an (induced) subgraph of G_2. That is, one has to find an injective mapping $f\colon V_1 \to V_2$ such that u and v are adjacent in G_1 if and only if $f(u)$ and $f(v)$ are adjacent in G_2.

Let $\mathcal{A}$ be a class of structures with two disjoint binary relations E^+ and E^- whose union is a full binary relation on the domain. The intended meaning is that

E^+ is the set of edges of a graph and E^- is the set of non-edges. It is not difficult to see that $\mathrm{CSP}(\mathcal{A}, \mathcal{A})$ corresponds to the subgraph isomorphism problem.

Structural restrictions. One type of restrictions are so called *structural* restrictions to the CSP. For a fixed class of relational structures $\mathcal{A}$, the problem $\mathrm{CSP}(\mathcal{A}, -)$ consists of those CSP instances $\mathcal{I}$ for which the structure $\mathbb{A}_\mathcal{I}$ belongs to $\mathcal{A}$. The idea is to restrict the structure that constraints induce on variables.

Example 3. Let $\mathcal{A}$ be the class of planar graphs. Then every instance of the graph 3-coloring problem that belongs to $(\mathcal{A}, -)$ asks about a coloring of a planar graph.

Purely structural restrictions to the classical constraint satisfaction problem are well understood and they can be summarised in the following result (the left-to-right implication holds under an additional assumption that $\mathrm{FPT} \neq \mathrm{W}[1]$ which is a standard hypothesis of parameterised complexity, cf. [22, 24]):

Theorem 1 (Dalmau, Kolaitis, Vardi [21]; Grohe [26])**.** *Let $\mathcal{A}$ be a recursively enumerable class of structures of bounded arity. Then $\mathrm{CSP}(\mathcal{A}, -)$ is polynomial-time solvable if and only if $\mathcal{A}$ has bounded tree-width modulo homomorphic equivalence.*

Language restrictions. In this thesis we focus on *language* restrictions to the CSP. That is, we consider the problem $\mathrm{CSP}(-, \mathcal{B})$ for a fixed class of structures $\mathcal{B}$. More precisely, most of the time we talk about so-called *non-uniform* CSPs [23], where $\mathcal{B} = \{\mathbb{B}\}$ consists of a single relational structure called a *constraint language* or a *template*. Instead of $\mathrm{CSP}(-, \{\mathbb{B}\})$ we write $\mathrm{CSP}(\mathbb{B})$. An instance of $\mathrm{CSP}(\mathbb{B})$ is a CSP instance over a domain B where every relation in every constraint is a relation from $\mathbb{B}$.

Example 4 (3-coloring)**.** Let $\mathbb{B}$ be a structure with three elements {red, blue, yellow}, and a single binary relation which is the inequality relation over the domain. Then $\mathrm{CSP}(\mathbb{B})$ is essentially the 3-coloring problem.

Example 5 (2-SAT)**.** Let $\mathbb{B}$ be a structure with two elements $\{0, 1\}$, and four binary relations: there is a relation $R_{(x,y)} = \{0, 1\}^2 \setminus \{(x, y)\}$, for each pair $(x, y) \in \{0, 1\}^2$. Then $\mathrm{CSP}(\mathbb{B})$ is the 2-SAT problem. For instance, a constraint $((z, t), R_{(0,0)})$ corresponds to a clause $z \vee t$.

The main goal is to classify non-uniform CSPs with respect to complexity. The first result of this form comes from Schaefer [45], who proved a complexity

dichotomy for boolean CSPs: if the domain of $\mathbb{B}$ has two elements then $\mathrm{CSP}(\mathbb{B})$ is either polynomial-time solvable, or NP-complete. Schaefer also provided a characterisation of those structures that give rise to polynomial-time solvable CSPs.

The famous *Dichotomy Conjecture* of Feder and Vardi states that the same complexity dichotomy holds for all constraint languages $\mathbb{B}$. That is, every non-uniform CSP is either tractable or NP-complete.

Conjecture 1 (Dichotomy Conjecture)**.** *For every relational structure $\mathbb{B}$ the problem $\mathrm{CSP}(\mathbb{B})$ is either in PTime or NP-complete.*

It is known [38] that if NP $\neq$ PTime then there are problems of intermediate complexity (problems in NP that are neither in PTime nor NP-complete). Proving the conjecture of Feder and Vardi would establish non-uniform CSPs as one of the biggest natural classes of computational problems to exhibit such a dichotomy.

Algebraic approach. A breakthrough in studying the complexity of the CSP came with an introduction of the so-called *algebraic approach* to constraint satisfaction. The remainder of this section presents the key ideas and developments of the algebraic approach, as many results of this thesis are based on this framework.

An *algebraic signature* Δ is a set of function symbols. Each symbol comes with a (finite) arity. An *algebra* $\mathbf{A}$ over Δ is a set A called a *universe* together with a set of functions called *basic operations* of $\mathbf{A}$. For every symbol f in Δ of arity n, there is a function $f^A\colon A^n \to A$ in $\mathbf{A}$. The algebraic approach to the CSP began with an observation that the complexity of $\mathrm{CSP}(\mathbb{B})$ can be characterised by the algebra of so-called *polymorphisms* of $\mathbb{B}$.

Definition 2 (Polymorphism of a relation)**.** Let R be a relation over a set B. A function $f\colon B^n \to B$ is a *polymorphism* of R if for any tuples $\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n \in R$ we have $f(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n) \in R$ (where f is applied component-wise). Then we also say that R is *compatible* with f.

Definition 3 (Polymorphism of a structure)**.** Let $\mathbb{B}$ be a relational structure with a domain B. A function $f\colon B^n \to B$ is a *polymorphism* of $\mathbb{B}$ if it is a polymorphism of every relation in $\mathbb{B}$. The set of polymorphisms of $\mathbb{B}$ is denoted $\mathrm{Pol}(\mathbb{B})$.

Example 6. A *majority* polymorphism of a relational structure $\mathbb{B}$ is a ternary polymorphism which for each pair x, y of elements of its domain, satisfies

$$m(x, x, y) = m(x, y, x) = m(y, x, x) = x.$$

If the domain is $\{0, 1\}$ then the above equations define m uniquely. Moreover, that unique m is a polymorphism of every binary relation R on $\{0, 1\}$. Indeed, take $\bar{x}_1, \bar{x}_2, \bar{x}_3 \in R$ and apply m component-wise:

$$
\begin{array}{lll}
\bar{x}_1 = & (x_1^1, x_1^2) & \in R \\
\bar{x}_2 = & (x_2^1, x_2^2) & \in R \\
\bar{x}_3 = & (x_3^1, x_3^2) & \in R \\
\hline\hline
 & \big(m(x_1^1, x_2^1, x_3^1), m(x_1^2, x_2^2, x_3^2)\big) & \in R
\end{array}
$$

It is not difficult to see that the tuple $\big(m(x_1^1, x_2^1, x_3^1), m(x_1^2, x_2^2, x_3^2)\big)$ is equal to one of the tuples $\bar{x}_1, \bar{x}_2, \bar{x}_3$. Hence, it belongs to R. Moreover, any unary relation is compatible with m. It follows that any boolean constraint language with relations of arity at most two has a majority polymorphism. This applies, in particular, to the template for 2-SAT (see Example 5).

With a relational structure $\mathbb{B}$ one can associate an algebra with universe B and a set of operations $\mathrm{Pol}(\mathbb{B})$. We call it the *polymorphism algebra* of $\mathbb{B}$. It was shown [33] that if ($\mathbb{A}$ and $\mathbb{B}$ have the same domain and) $\mathrm{Pol}(\mathbb{B}) \subseteq \mathrm{Pol}(\mathbb{A})$ then $\mathrm{CSP}(\mathbb{A})$ is polynomial-time reducible to $\mathrm{CSP}(\mathbb{B})$. In particular, if $\mathrm{Pol}(\mathbb{A}) = \mathrm{Pol}(\mathbb{B})$ then $\mathrm{CSP}(\mathbb{A})$ and $\mathrm{CSP}(\mathbb{B})$ are polynomial-time equivalent. This means that indeed the complexity of non-uniform CSPs can be analysed via their corresponding algebras of polymorphisms, which allows to use powerful results from universal algebra.

Of particular importance in universal algebra are *idempotent* algebras, i.e., algebras whose every operation f satisfies $f(x, \ldots, x) = x$, for every element x of the universe (such an operation is also called *idempotent*). It was shown [15, 14] that while studying the complexity of the CSP one can restrict attention to idempotent algebras. This is due to the fact that for every relational structure $\mathbb{A}$ there exists a *core* structure $\mathbb{B}$ such that $\mathrm{CSP}(\mathbb{A})$ and $\mathrm{CSP}(\mathbb{B})$ are polynomial-time equivalent. A *core* is a relational structure whose all unary polymorphisms are bijective. To a core structure $\mathbb{B}$ one can add all one-element unary relations without increasing the complexity. To sum up, for every relational structure $\mathbb{A}$ there exists a relational structure $\mathbb{B}$, such that $\mathrm{CSP}(\mathbb{A})$ and $\mathrm{CSP}(\mathbb{B})$ are computationally equivalent, and the structure $\mathbb{B}$, for every element b of its domain, contains the relation $\{b\}$. It is easy to see that every polymorphism of $\mathbb{B}$ then is idempotent.

A *term* over a signature Δ is defined recursively: a variable is a term; and if $t_1, \ldots, t_n$ are terms, and $f \in \Delta$ is an n-ary function symbol, then $f(t_1, \ldots, t_n)$ is a term. For an algebra $\mathbf{A}$ over Δ, a *term operation* $t^{\mathbf{A}}$ is a function obtained by composing the basic operations of $\mathbf{A}$ according to t. If s and t are terms then we

say that $\mathbf{A}$ satisfies the *identity* $s \approx t$ if $s^{\mathbf{A}} = t^{\mathbf{A}}$. The next step in the algebraic approach to the CSP was to move from considering single algebras to studying *varieties*, which are classes of algebras (over the same signature) defined by sets of identities.

Definition 4 (Variety)**.** For any set of identities I over a signature Δ, a *variety* $\mathcal{V}$ is the class of all algebras over Δ that satisfy every identity from I.

There is a different way of introducing varieties, where a variety is defined as a class of algebras closed under certain operations (see e.g. [16]). The equivalence of both definitions was shown by Birkhoff [5].

By $\mathcal{V}(\mathbf{B})$ we denote a variety *generated* by an algebra $\mathbf{B}$, i.e., the class of algebras that satisfy all identities satisfied by $\mathbf{B}$. Let $\mathbf{B}$ be the polymorphism algebra of $\mathbb{B}$, and let $\mathbf{A}$ be some algebra in $\mathcal{V}(\mathbf{B})$. It turns out [15, 14, 39] that $\mathrm{CSP}(\mathbb{A})$, where $\mathbb{A}$ is a relational structure whose domain is the universe of $\mathbf{A}$, and whose relations are compatible with operations in $\mathbf{A}$, is no harder than $\mathrm{CSP}(\mathbb{B})$. Informally, this means that the complexity of a non-uniform CSP can be captured by a set of identities.

The *Algebraic Dichotomy Conjecture* of Bulatov, Jeavons, and Krokhin [15, 14] makes the conjecture of Feder and Vardi more specific by proposing a concrete algebraic condition to distinguish tractable and intractable CSPs. For a core structure $\mathbb{B}$, they conjecture $\mathrm{CSP}(\mathbb{B})$ to be NP-complete if in the variety generated by its polymorphism algebra there is a two-element algebra whose every operation is a projection, and solvable in PTime otherwise.

Further work lead to an equivalent formulation of the conjecture, which uses the notion of a *cyclic* polymorphism. A polymorphism f of arity $n \geq 2$ is called *cyclic* if it satisfies the identity $f(x_1, x_2, \ldots, x_n) \approx f(x_2, \ldots, x_n, x_1)$. Barto and Kozik [3] showed that the hardness condition in the conjecture above is equivalent to saying that $\mathbb{B}$ does not have an idempotent cyclic polymorphism.

Conjecture 2 (Algebraic Dichotomy Conjecture)**.** *Let $\mathbb{B}$ be a relational structure that is a core. If $\mathbb{B}$ does not have an idempotent cyclic polymorphism, then $\mathrm{CSP}(\mathbb{B})$ is NP-complete. Otherwise it is polynomial-time solvable.*

While stating their conjecture, Bulatov et al. proved its one direction: they showed that if in the variety generated by the polymorphism algebra of a core structure $\mathbb{B}$ there is a two-element algebra whose every operation is a projection then $\mathrm{CSP}(\mathbb{B})$ is NP-complete. Using the characterisation with cyclic polymorphism their result can be formulated as follows:

Theorem 2 (Bulatov, Jeavons, Krokhin [15, 14])**.** *Let $\mathbb{B}$ be a relational structure that is a core. If $\mathbb{B}$ does not have an idempotent cyclic polymorphism, then $\mathrm{CSP}(\mathbb{B})$ is NP-complete.*

The Algebraic Dichotomy Conjecture agrees with the result of Schaefer for boolean CSPs. It was also confirmed for structures over three-element domains [12], and for so-called *conservative* languages [11].

Even though the Dichotomy Conjecture is still open, the algebraic approach resulted in a remarkable progress in understanding the complexity of the CSP. An interesting direction is characterising constraint satisfaction problems tractable in a stronger sense than polynomial-time solvability. Of particular interest in this thesis is definability in Datalog – a logic obtained by extending unions of conjunctive queries by a fixpoint operator (cf. [1]).

Bounded width. Probably the most well-known polynomial-time algorithm for solving constraint satisfaction problems is the *consistency algorithm* with parameters k and l (where $k \leq l$). We recall it here after Barto and Kozik [4] (by $\mathrm{dom}\, f$ we denote the domain of a function f):

Input: An instance $\mathcal{I} = (V, D, \mathcal{C})$
Output: YES/NO
$\mathcal{F} =$ all functions from at most l-element subsets of V into D;
  for $f \in \mathcal{F}$ **do**
    for $((x_1, \ldots, x_n), R) \in \mathcal{C}$ **do**
      if $x_1, \ldots, x_n \in \mathrm{dom}\, f$ and $(f(x_1), \ldots, f(x_n)) \notin R$ **then**
        $\mathcal{F} = \mathcal{F} \setminus \{f\}$;
        break;
repeat
  for $f \in \mathcal{F}$ **do**
    foreach W at most l-element subset of V **do**
      if ($|\mathrm{dom}\, f| \leq k, \mathrm{dom}\, f \subseteq W$
        and there is no $g \in \mathcal{F}$ with $\mathrm{dom}\, g = W$ and $g|_{\mathrm{dom}\, f} = f$)
        or ($W \subseteq \mathrm{dom}\, f$ and $f|_W \notin \mathcal{F}$) **then**
          $\mathcal{F} = \mathcal{F} \setminus \{f\}$;
          break;
until $\mathcal{F}$ was not altered;
if $\mathcal{F} = \emptyset$ **then return** NO **else return** YES

Intuitively, the (k, l)-consistency algorithm constructs a family of partial solutions of the instance (defined on all sets of at most l variables) which are pair-wise

consistent on sets of at most k variables. If an instance has a solution then such a family always exists and the algorithm's answer is correct ('YES'). We say that a template $\mathbb{B}$ has *bounded width* if there exist numbers $k \leq l$ such that the (k, l)-consistency algorithm provides a correct answer ('NO') for every unsatisfiable instance of $\mathrm{CSP}(\mathbb{B})$. Examples include templates for 2-colorability and 2-SAT.

Constraint languages of bounded width were introduced in the seminal paper of Feder and Vardi [23]. The original (equivalent) definition says that a structure $\mathbb{B}$ has bounded width if the set of structures which do not map homomorphically into $\mathbb{B}$ is definable in Datalog. The (k, l)-consistency checking can be seen as a canonical Datalog program.

Constraint satisfaction problems solvable in Datalog form a very robust class, with many equivalent characterisations. One of them, using the notion of so-called *ability to count*, was suggested already in [23]. We say that a template $\mathbb{B}$ has the *ability to count* if $\mathrm{CSP}(\mathbb{B})$ can simulate solving systems of linear equations over a finite field (for a precise definition see [40]). In [23] it was shown that constraint languages with the ability to count fail to have bounded width. In the same paper Feder and Vardi conjectured that solving linear equations is essentially the only obstacle to bounded width.

Example 7. A classical example of a template which has the ability to count is a template for solving linear equations over the two-element field $\mathbb{Z}_2$, with three variables per equation. Its domain is $\{0, 1\}$, and there are two relations that encode solutions to two possible kinds of equations with three variables (by $+$ we denote addition modulo 2):

$$R_1 = \{(x, y, z) \mid x + y + z = 1\},$$
$$R_0 = \{(x, y, z) \mid x + y + z = 0\}.$$

By the result of Feder and Vardi [23] this template does not have bounded width.

Another famous conjecture regarding bounded width is one by Larose and Zádori [41] who proposed a characterisation in terms of algebra. A polymorphism f is a *weak near unanimity* polymorphism if it is idempotent and satisfies the identities

$$f(y, x, \ldots, x) \approx f(x, y, x, \ldots, x) \approx \ldots \approx f(x, x, \ldots, x, y).$$

Larose and Zádori proved that if a core template $\mathbb{B}$ has bounded width then it

has weak near unanimity polymorphisms of all but finitely many arities[1], and conjectured this algebraic condition to be also sufficient.

In [40] the above conjectures were shown to be equivalent, to finally be confirmed by Barto and Kozik [4], and independently by Bulatov [13]. Using additionally a recent development of [37] their result can be stated as follows:

Theorem 3. *Let $\mathbb{B}$ be a core structure. Then $\mathbb{B}$ has bounded width if and only if $\mathbb{B}$ has a pair of weak near unanimity polymorphisms v and w of arity 3 and 4, respectively, satisfying $v(y,x,x) \approx w(y,x,x,x)$.*

The advantage of the above formulation lies in the fact that the required polymorphisms have bounded arity. For a given structure $\mathbb{B}$ there are finitely many candidates for polymorphisms of a fixed arity. Their existence can be verified by an algorithm. Therefore, the characterisation is effective.

2. Sets with atoms

The classical constraint satisfaction problem and the algebraic perspective presented above concern finite CSP instances. In this thesis we are also interested in the case when the instances are infinite. Sets with atoms provide a convenient framework for modeling infinite objects and representing them in a finite way so that they can be treated as input for algorithms.

Sets with atoms were introduced by Fraenkel in 1922 as an alternative set theory. Just as classical sets are built of empty sets and brackets, sets with atoms might also contain so-called *atoms*, which do not have any elements themselves. Axioms of the set theory with atoms (call it ZFA) are similar to the Zermelo-Fraenkel axiomatisation of the standard set theory. There are, however, some differences. For example, the axiom of extensionality is weakened to allow two different atoms to have the same elements (namely, no elements) and not be equal. ZFA was originally introduced to show independence of some axioms from others: it has models where the axiom of choice fails. In this context sets with atoms were further studied by Mostowski, which is why they are sometimes called Fraenkel-Mostowski sets.

Sets with atoms were rediscovered for computer science by Gabbay and Pitts, under the name of *nominal sets* [25, 44]. They received considerable attention

[1]The original statement in [41] uses Tame Congruence Theory (TCT) [28]. The version we present here is based on a result from [42] where varieties omitting TCT-types 1 and 2 are characterised by weak near unanimity terms.

in the semantics community where they are used to model name-binding and α-conversion (for details see [44]). This application area is not of concern in the present thesis but the notions below are based on [25, 44], with some extensions defined in [8, 6].

Atoms. The notion of sets with atoms is parametrised by a relational structure which we denote $\mathcal{A}toms$. We assume its signature to be finite, and its domain to be countable. An *atom* is an element of the domain, which for simplicity we also denote by $\mathcal{A}toms$. The most important examples of $\mathcal{A}toms$ are natural numbers with the equality relation $(\mathbb{N}, =)$, and rational numbers with order $(\mathbb{Q}, <)$.

A (left) *action* of a group G on a set X is a function that maps each pair $(\pi, x) \in G \times X$ to an element $\pi \cdot x$ of X, and satisfies $\pi\sigma \cdot x = \pi \cdot (\sigma \cdot x)$, and $1 \cdot x = x$, where 1 is the identity element of G. An *orbit* of an element $x \in X$ under the action of a group G is the set $G \cdot x = \{\pi \cdot x \mid \pi \in G\}$. The set X is then a disjoint union of orbits.

We shall be interested in the action of the automorphism group $\mathrm{Aut}(\mathcal{A}toms)$ of $\mathcal{A}toms$ on the set of n-tuples of atoms. The action is defined in a natural way: for an automorphism $\pi \in \mathrm{Aut}(\mathcal{A}toms)$ and an n-tuple $(a_1, \ldots, a_n)$ of atoms, we put $\pi \cdot (a_1, \ldots, a_n) = (\pi(a_1), \ldots, \pi(a_n))$. In this thesis we consider only *oligomorphic* atom structures, which means that the action of the automorphism group on the set of n-tuples has finitely many orbits for every n. The structures $(\mathbb{N}, =)$, and $(\mathbb{Q}, <)$ are oligomorphic. Oligomorphicity is a well-studied notion in model theory (see e.g. [29]).

Example 8. Fix $\mathcal{A}toms$ to be natural numbers with the equality relation. Then the set of pairs of atoms has two orbits:

$$\{(a, b) \mid a, b \in \mathbb{N},\ a \neq b\} \text{ and } \{(a, a) \mid a \in \mathbb{N}\}.$$

Now let $\mathcal{A}toms$ be rational numbers with order. Then the set of pairs of atoms has three orbits:

$$\{(a, b) \mid a, b \in \mathbb{Q},\ a < b\},\ \{(a, b) \mid a, b \in \mathbb{Q},\ a > b\}, \text{ and } \{(a, a) \mid a \in \mathbb{Q}\}.$$

Example 9. Consider the structure $(\mathbb{Z}, <)$ of integer numbers with order. This structure is not oligomorphic. To see this observe that the automorphism group of $(\mathbb{Z}, <)$ is the group of translations $i \mapsto i + c$, isomorphic to the additive group of integers. The set of pairs of integers has infinitely many orbits under the action of $\mathrm{Aut}(\mathbb{Z}, <)$. An orbit of a pair (m, n) is uniquely determined by the number $m - n$. Taking $\mathcal{A}toms$ to be the structure $(\mathbb{Z}, <)$ results in pathological properties of sets with atoms (see [8]).

We shall now introduce sets with atoms, the way they are defined in [25].

Set-theoretic approach. Given a structure $\mathcal{A}toms$, sets with atoms are defined by ordinal induction. The only set on level 0 is the empty set. Sets on level α are atoms, or contain sets at levels smaller than α. Examples of sets with atoms include:

- the set of atoms itself,
- the set $\mathcal{A}toms^n$ of n-tuples of atoms (where tuples are encoded by usual set-theoretic constructions),
- the set $\mathcal{A}toms^{(n)}$ of non-repeating n-tuples of atoms,
- the set $\mathcal{P}_{fin}(\mathcal{A}toms)$ of all finite sets of atoms,
- any classical set without atoms.

The group $\mathrm{Aut}(\mathcal{A}toms)$ acts on the family of sets on each level α. When we apply an automorphism π to a set with atoms X, we rename via π the atoms that are its elements, and the atoms that are elements of its elements, and so on recursively.

By $\mathrm{Aut}(\mathcal{A}toms, a_1, \ldots, a_n)$, where $a_i \in \mathcal{A}toms$, we denote the group of those automorphisms of $\mathcal{A}toms$ that fix all the a_i's.

Definition 5 (Support)**.** A set $S = \{a_1, \ldots, a_n\}$ of atoms is a *support* of a set with atoms X if $\pi \cdot X = X$ for every automorphism $\pi \in \mathrm{Aut}(\mathcal{A}toms, a_1, \ldots, a_n)$. Then we also say that X is *S-supported*.

A set with atoms is *equivariant* if it has an empty support. All sets with atoms mentioned above are equivariant. An example of a finitely supported set with atoms which is not equivariant is the set of all pairs of atoms with the atom a on the first coordinate. One of its finite supports is the set $\{a\}$.

The definition of a finitely supported set with atoms applies to relations and functions. Let X and Y be sets with atoms. A finitely supported relation is any finitely supported subset of $X \times Y$. A function $f\colon X \to Y$ is S-supported (equivariant) if its graph is S-supported (respectively equivariant) as a subset of $X \times Y$. Equivalently, for a finite set $S = \{a_1, \ldots, a_n\}$ of atoms, the function f is S-supported if $f(\pi \cdot x) = \pi \cdot f(x)$, for every $x \in X$, and every automorphism $\pi \in \mathrm{Aut}(\mathcal{A}toms, a_1, \ldots, a_n)$.

The following fact (see [6]) provides a useful representation of finitely supported subsets of $\mathcal{A}toms^k$, for oligomorphic atom structures.

Proposition 4. *If the structure $\mathcal{A}tom s$ is oligomorphic then a set of k-tuples of atoms is S-supported if and only if it is definable by a first order formula over the signature of $\mathcal{A}tom s$ with k free variables and constants from S.*

A set with atoms is *hereditarily finitely supported* if it has a finite support, and all its elements have finite supports, and so on. In the following we consider *only hereditarily finitely supported sets with atoms*, so we omit this qualification.

Example 10. Fix $\mathcal{A}tom s$ to be natural numbers with the equality relation. Then it is not difficult to see that the finitely supported subsets of atoms are exactly finite sets of atoms and those whose complement is finite. Therefore, the set $\mathcal{P}(\mathcal{A}tom s)$ of all subsets of atoms is not hereditarily finitely supported. On the other hand, the set $\mathcal{P}_{fin}(\mathcal{A}tom s)$ of all finite subsets of atoms is hereditarily finitely supported.

Let X be a set with atoms supported by $S = \{a_1, \ldots, a_n\}$. Then the group $\mathrm{Aut}(\mathcal{A}tom s, a_1, \ldots, a_n)$ acts on X in the standard way (by renaming the atoms that appear in elements of X). An orbit of this action is called an S-orbit. If X has finitely many S-orbits then X is called *orbit-finite with respect to S*. Notice that a set with atoms does not have a unique finite support. In particular, supports are closed under adding atoms. For oligomorphic atoms, it can be shown (see [6]) that if X is orbit-finite with respect to some finite support S then it is orbit-finite with respect to any of its finite supports. Therefore, we call it simply *orbit-finite*.

Example 11. For oligomorphic atoms the set of atoms, and the set of n-tuples of atoms are orbit-finite. The set of all finite subsets of atoms is not orbit-finite, as any two subsets of different cardinality are in different orbits.

To treat sets with atoms as an input for computation we need finite means to represent them. Below we introduce the finite representation that we use in [P1]. Different approaches can be found e.g. in [18, 19, 8].

Set expressions. A *definable* set with atoms is specified by a *set expression* together with a *valuation* of its free variables. A *set expression* is defined inductively. It can be:

- a variable from some fixed infinite set of variables Var,
- an integer,
- a formal tuple of set expressions,
- a formal union of set expressions,

- a *set-builder expression* of the form

$$\{e(\bar{x}\bar{y}) \mid \bar{y} \in \mathcal{A}toms^n,\ \phi(\bar{x}\bar{y})\},$$

  where $\bar{x}$ is an n-tuple of free variables, $\bar{y}$ is a k-tuple of bound variables, e is an already defined set expression with $n+k$ free variables, and ϕ is a first order formula over $\mathcal{A}toms$ with $n+k$ free variables.

We allow only set expressions where the formal union is applied to set-builder expressions or unions of set-builder expressions.

The formal tuple of set expressions is not necessary in the definition above. It is introduced to simplify the syntax. We fix some set-theoretic encoding of tuples and treat the formal tuple as a shorthand for the set expression describing this encoding.

The integers are also introduced to improve readability and could be got rid of. We fix some set-theoretic encoding of integers, where an integer is represented by a hereditarily finite set (without atoms)[2]. In the definition of a set expression all the integers except for 0 can be removed. If we use the constant 0 to represent the empty set then the encoding of any integer can be defined using standard set-theoretic construction.

A *valuation* of the free variables $\bar{x}$ is a function that assigns an atom to every variable in the tuple $\bar{x}$. A set expression together with a valuation defines a set with atoms in an obvious way. The semantics is very intuitive and we shall not give its precise description. Below we show that any definable set with atoms is orbit-finite. First let us have a look at some examples.

Example 12. Examples of definable sets with atoms include:

- an atom a, as defined by an expression x (a variable) together with a valuation $x \mapsto a$,
- the sets of non-repeating pairs of atoms, as defined by an expression

$$\{(y_1, y_2) \mid (y_1, y_2) \in \mathcal{A}toms^2,\ y_1 \neq y_2\}.$$

Proposition 5. *A set expression together with a valuation defines an orbit-finite set with atoms.*

[2]For natural numbers we can use the von Neumann construction (see e.g. [30]). Then a negative number $-n$ can be encoded as the set that represents n with some additional special element.

Proof. We show this by induction on the structure of the set expression. Recall that a formal tuple of set expressions is only syntactic sugar, so we do not need to take it into account in the proof.

A variable x together with a valuation $x \mapsto a$ defines an atom a which is obviously orbit-finite, since it has no elements.

A set expression which is an integer has no free variables. It defines its set-theoretical encoding as a classical set without atoms. We fixed the encoding to be a finite set, hence it is orbit-finite (each element is in a different orbit).

Now consider a formal union of set expressions $e_1(\bar{x}_1)$ and $e_2(\bar{x}_2)$ together with some valuation val of the free variables in the tuples $\bar{x}_1$ and $\bar{x}_2$. By the inductive assumption the set expression $e_1(\bar{x}_1)$ together with the valuation val restricted to the free variables in $\bar{x}_1$, and the set expression $e_2(\bar{x}_2)$ together with the valuation val restricted to the free variables in $\bar{x}_2$ are orbit-finite sets with atoms X_1 and X_2, respectively. Recall that we only allow the formal union to be applied to set-builder expressions and unions of set-builder expressions. Hence, we can assume that neither X_1 nor X_2 is an atom or an integer. Then the formal union of $e_1(\bar{x}_1)$ and $e_2(\bar{x}_2)$ defines the union of X_1 and X_2 which is orbit-finite (with respect to the sum of supports of X_1 and X_2).

Finally, let $e(\bar{x})$ be a set-builder expression of the form

$$\{e^*(\bar{x}\bar{y}) \mid \bar{y} \in \mathcal{A}toms^n,\ \phi(\bar{x}\bar{y})\},$$

where $\bar{x}$ and $\bar{y}$ are disjoint tuples of variables of length n and k, respectively. And let val be some valuation of the free variables in the tuple $\bar{x}$. Then $e(\bar{x})$ together with val defines a set with atoms X. We need to show that X is orbit-finite. Let $S = \{a_1, \ldots, a_n\}$ be the image of val. In the first order formula $\phi(\bar{x}\bar{y})$ we replace the free variables in $\bar{x}$ by constants from S according to val, and we obtain a formula $\phi'(\bar{y})$. By Proposition 4, the formula $\phi'(\bar{y})$ defines an S-supported set Y of k-tuples of $\mathcal{A}toms$. Since the atoms are oligomorphic, it is not difficult to see that Y is orbit-finite.

Now the set expression $e(\bar{x})$ can be (informally) rewritten as follows:

$$\{e^*(\bar{x}\bar{y}) \mid \bar{y} \in Y\}.$$

By this we mean that each element of X is a set with atoms defined by the set expression $e^*(\bar{x}\bar{y})$ together with a valuation which maps the tuple of free variables $\bar{x}$ to a tuple of atoms according to val, and the tuple of free variables in $\bar{y}$ to some tuple of atoms in Y. Hence, elements of X are determined by k-tuples in Y.

Take k-tuples $\bar{b}_1$ and $\bar{b}_2$ that belong to the same S-orbit of Y. Let val_1 and val_2 be extensions of the valuation val which additionally map the variables in the tuple $\bar{y}$ to the corresponding atoms in the tuples $\bar{b}_1$ and $\bar{b}_2$, respectively. The set expression $e^*(\bar{x}\bar{y})$ together with each of the valuations val_1 and val_2 defines elements of X. We denote them by x_1 and x_2, respecitively. There exists $\pi \in \mathrm{Aut}(\mathcal{A}toms, a_1, \ldots, a_n)$ which maps the tuple $\bar{b}_1$ to the tuple $\bar{b}_2$. It is not difficult to show that such a π maps x_1 to x_2. Hence, x_1 and x_2 are in the same S-orbit. Since there are finitely many S-orbits in Y, it follows that there are finitely many S-orbits in X, which finishes the proof. □

Both in [P1] and [P2] we work within the framework of sets with atoms but in [P1] we confine to those that are definable. Below we show that every orbit-finite set with atoms is related by a finitely supported bijection to a definable set with atoms. Together with Proposition 5 this implies that orbit-finite and definable sets with atoms are essentially equivalent.

As a first step we prove that orbit-finite sets with atoms can be represented as sets of tuples of atoms modulo a first-order definable equivalence relation. We need an auxiliary lemma.

Lemma 6. *Let $\sim$ be an S-supported equivalence relation on a set with atoms X, then the quotient $X/\!\sim$ is an S-supported set with atoms.*

Proof. Let $S = \{a_1, \ldots, a_n\}$. Observe that X is an S-supported set itself. Take $\pi \in \mathrm{Aut}(Atoms, a_1, \ldots, a_n)$. We need to show that $\pi \cdot X/\!\sim = X/\!\sim$. Since $\sim$ is an S-supported relation, for every $x \in X$, we have $\pi \cdot [x]_\sim \subseteq [\pi \cdot x]_\sim$. Moreover, the mapping $x \mapsto \pi \cdot x$ is bijective. Since the equivalence classes of $\sim$ form a partition of the set X, it follows that indeed $\pi \cdot X/\!\sim = X/\!\sim$.

It remains to show that $X/\!\sim$ is a "legal" set with atoms, i.e., that it is hereditarily finitely supported. We already know that $X/\!\sim$ is finitely supported. Take an equivalence class $[x]_\sim$. Since x is an element of X it has some finite support $T = \{b_1, \ldots, b_l\}$. Let $\pi \in \mathrm{Aut}(\mathcal{A}toms, a_1, \ldots, a_n, b_1, \ldots, b_l)$. Then $\pi \cdot [x]_\sim = [\pi \cdot x]_\sim = [x]_\sim$. Hence, the set $[x]_\sim$ is supported by $S \cup T$. Elements of the equivalence classes, and elements of those elements, and so on, are finitely supported, since X is hereditarily finitely supported. □

Proposition 7. *Let X be an S-supported set which is a single S-orbit. Then X is in an S-supported bijection with a set $\mathcal{O}/\!\sim$, where $\mathcal{O}$ is an S-orbit of $\mathcal{A}toms^m$, and $\sim$ is an equivalence relation definable by a first order formula with $2m$ free variables and constants from S.*

Proof. Take some element x of X, and let $\bar{u}$ be an ordered tuple of all elements in some support of x. Let $\mathcal{O} \subseteq \mathcal{A}toms^m$ be the S-orbit of the tuple $\bar{u}$. We consider a function $f\colon \mathcal{O} \to X$ which for $\pi \in \mathrm{Aut}(Atoms, a_1, \ldots, a_n)$, maps $\pi \cdot \bar{u}$ to $\pi \cdot x$. This function is well-defined. Indeed, if $\pi \cdot \bar{u} = \sigma \cdot \bar{u}$, then $\sigma^{-1}\pi \cdot \bar{u} = \bar{u}$. Since the set of atoms in $\bar{u}$ supports x, this implies that $\sigma^{-1}\pi \cdot x = x$, and $\pi \cdot x = \sigma \cdot x$. Moreover, it follows from the definition of f that it is supported by S. Let $\sim$ be the kernel of f, i.e., an equivalence relation where $\bar{v} \sim \bar{w}$ if and only if $f(\bar{v}) = f(\bar{w})$. It is not difficult to see that the relation $\sim$ is also S-supported. Therefore, by Lemma 6 the quotient $\mathcal{O}/\!\sim$ is an S-supported set with atoms. The function f induces an S-supported bijection between $\mathcal{O}/\!\sim$ and X. The equivalence class $[\bar{v}]_\sim$ is mapped to $f(\bar{v})$.

It remains to show that $\sim$ is first-order definable. To do this we observe that $\sim$ is an S-supported subset of $\mathcal{A}toms^m \times \mathcal{A}toms^m$. This follows from the fact that f is supported by S. Indeed, if $\bar{v} \sim \bar{w}$, then $f(\bar{v}) = f(\bar{w})$, hence for every automorphism $\pi \in \mathrm{Aut}(\mathcal{A}toms, a_1, \ldots, a_n)$ we have $f(\pi \cdot \bar{v}) = f(\pi \cdot \bar{w})$, therefore $\pi \cdot \bar{v} \sim \pi \cdot \bar{w}$. By Proposition 4, the equivalence relation $\sim$ is definable by a first order formula with $2m$ free variables and constants from S. □

Example 13. Let $\mathcal{A}toms$ be natural numbers with equality. The set of two-element subsets of $\mathcal{A}toms$ is related by an equivariant bijection to the set $\mathcal{O}/\!\sim$, where $\mathcal{O}$ is the single orbit of pairs of distinct atoms, and for every two distinct atoms a, b we have $(a, b) \sim (b, a)$, i.e., $\sim$ is defined by the formula

$$(x_1 = y_1 \wedge x_2 = y_2) \vee (x_1 = y_2 \wedge x_2 = y_1).$$

Theorem 8. *Every orbit-finite set with atoms is in a finitely supported bijection with a definable set with atoms.*

Proof. First, let us deal with the special case of single-orbit sets. Let X be an S-supported set which is a single S-orbit, where $S = \{a_1, \ldots, a_n\}$. By Proposition 7 X is related by an S-supported bijection to a set $\mathcal{O}/\!\sim$, where $\mathcal{O}$ is an S-orbit of $\mathcal{A}toms^m$, and $\sim$ is an equivalence relation definable by a first order formula with $2m$ free variables and constants from S. We show that $\mathcal{O}/\!\sim$ is a definable set with atoms.

Let $\bar{x}$ and $\bar{y}$ be m-tuples of variables, and let $\varphi(\bar{x}\bar{y})$ be a first order formula that defines $\sim$. Let $\bar{a}$ be an m-tuple of atoms. A set expression

$$e^*(\bar{y}) = \{\bar{x} \mid \bar{x} \in \mathcal{A}toms^m,\ \varphi(\bar{x}\bar{y})\}$$

with m free variables together with a valuation that maps the variables in the tuple $\bar{y}$ to the corresponding atoms in the tuple $\bar{a}$ defines the equivalence class of $\bar{a}$.

The set $\mathcal{O}/\sim$ consists of equivalence classes of tuples in $\mathcal{O}$. Let $\psi(\bar{y})$ be a first order formula with m free variables that defines $\mathcal{O}$. Then the quotient $\mathcal{O}/\sim$ is defined by

$$e = \{e^*(\bar{y}) \mid \bar{y} \in \mathcal{A}toms^m,\ \psi(\bar{y})\}.$$

According to the definition, a set expression cannot use constants. Therefore, we have to slightly modify e. In the formulas φ and ψ every constant a_i from S has to be replaced with a free variable z_i. The new expression has an n-tuple $\bar{z}$ of free variables. Together with a valuation which maps every z_i to a_i it defines $\mathcal{O}/\sim$.

Now let X be any orbit-finite set with atoms supported by $S = \{a_1, \ldots, a_n\}$; then X is a disjoint union of finitely many S-orbits $X_1, \ldots, X_l$. Above we showed that for every $i \in \{1, \ldots, l\}$, the orbit X_i is related by an S-supported bijection to a set X_i' definable by a set expression

$$\{e_i^*(\bar{y}\bar{z}) \mid \bar{y} \in \mathcal{A}toms^{m_i},\ \psi_i(\bar{y}\bar{z})\}$$

together with some valuation val_i. The problem is that the sets X_i' might not be disjoint. Therefore, instead of each X_i' we take $X_i' \times \{i\}$ which is definable by a set expression

$$e_i(\bar{z}) = \{\big(e_i^*(\bar{y}\bar{z}), i\big) \mid \bar{y} \in \mathcal{A}toms^{m_i},\ \psi_i(\bar{y}\bar{z})\},$$

where $\big(e_i^*(\bar{y}\bar{z}), i\big)$ is a formal tuple of a set expression $e_i^*(\bar{y}\bar{z})$ and an integer i. Then the set X is related by an S-supported bijection to a set with atoms defined by a set expression which is a formal union of the set expressions $e_i(\bar{z})$ together with a valuation induced by the valuations val_i (we make sure that the sets of free variables in the expressions e_i are disjoint). □

It should be stressed that the correspondence between orbit-finite and definable sets with atoms crucially relies on $\mathcal{A}toms$ being an oligomorphic structure.

Example 14. Take $\mathcal{A}toms$ to be integer numbers with order. Recall from Example 9 that the set of pairs of integers has infinitely many orbits under the action of $\mathrm{Aut}(\mathbb{Z}, <)$. However, the set of pairs of atoms is obviously definable by a set expression $\{(y_1, y_2) \mid (y_1, y_2) \in \mathcal{A}toms\}$.

Relation to nominal sets. In the definition of a nominal set, as given in [25], atoms are assumed to have no structure except for equality, i.e., $\mathcal{A}toms$ are fixed to be $(\mathbb{N}, =)$. A nominal set is a set X together with an action of the group of

all permutations of atoms, such that every element of X is finitely supported. It follows from results of Ciancia and Montanari [18, 19] that every orbit-finite nominal set is related by a finitely supported bijection to an orbit-finite set with atoms.

3. Overview of the results

Having introduced the key notions of the constraint satisfaction paradigm, and the language of sets with atoms, we shall now give a summary of the most important results of the three papers which constitute this thesis.

3.1. Deciding definable CSP instances [P1]

In [P1] we study definable CSP instances, where the set of variables, the domain, and the set of constraints are definable sets with atoms. Most of the time $\mathcal{A}toms$ are fixed to be natural numbers with the equality relation.

Example 15. Consider the following instance of 3-colorability. The graph has infinitely many vertices which are pairs of distinct atoms, i.e., the set of variables is defined by the set expression $\{(a, b) \mid a, b \in \mathcal{A}toms,\ a \neq b\}$. The domain is the set of possible colors, say $\{0, 1, 2\}$. For every triple of distinct atoms a, b, c there is an edge between the vertices (a, b) and (b, c). So the set of constraints is defined by

$$\{\big(((a, b), (b, c)), R\big) \mid a, b, c \in \mathcal{A}toms,\ a \neq b \wedge b \neq c \wedge a \neq c\},$$

where R is the expression $\{(0, 1), (0, 2), (1, 0), (1, 2), (2, 0), (2, 1)\}$.

In the above example the domain of the instance is finite. In general, we are interested in definable instances over possibly infinite templates. A definition of a template is the same as in the classical CSP, but now the template can be an infinite relational structure, i.e., its domain, as well as the number of relations might be infinite. We assume templates to be *locally finite*, which means that every relation in the structure is finite.

Example 16. In a generalised 2-colorability problem we ask for a coloring of a graph, but the set of possible colors is infinite. Every vertex is assigned a set of two colors (that come from the infinite set). As usual, each vertex has to be mapped to one of the two colors assigned to it, such that no two adjacent vertices share

the same color. This problem can be seen as a constraint satisfaction problem over a definable, locally finite template with infinitely many binary relations. The domain is the set of atoms defined by $\{a \mid a \in \mathcal{A}toms\}$, and the set of relations is

$$\{\{(a,c),(a,d),(b,c),(b,d)\} \mid a,b,c,d \in \mathcal{A}toms,\ a \neq b \wedge c \neq d\}.$$

This set of relations has three orbits. An orbit of a relation depends on how many colors the sets $\{a,b\}$ and $\{c,d\}$ have in common. If the sets of colors are equal then the relation contains two pairs. If the sets of colors have a one-element intersection then there are three pairs in the relation. If they are disjoint then the relation has four elements. The template for the generalised 2-colorability problem is therefore locally finite.

The main result of [P1] is decidability of definable CSP instances over locally finite templates.

Theorem 9. *For any definable, locally finite template $\mathbb{T}$, it is decidable whether a given definable instance $\mathcal{I}$ over $\mathbb{T}$ has a solution.*

In [P1] we prove the above theorem for the special case of equivariant templates and instances, and we indicate how our arguments can be generalised to cover the general case. In the sketch of the proof presented below we also implicitly restrict to equivariant templates and instances.

The idea is to show decidability of instances definable over $\mathcal{A}toms$ that are rational numbers with order. Then it remains to notice that instances definable by formulas which use the equality relation only are a special case of instances definable by formulas that use the order relation. To avoid confusion, whenever we talk about instances definable over $\mathcal{A}toms$ that are rational numbers with order we indicate this by writing *order-definable*. Moreover, sets equivariant with respect to the action of the group $\mathrm{Aut}(\mathbb{Q},<)$ we call *monotone-equivariant*.

Take an order-definable instance $\mathcal{I}$ over a locally finite template $\mathbb{T}$, and consider the space of all assignments of values to variables with the topology of pointwise convergence (cf. [35]). We notice that the space of solutions $\hom(\mathcal{I},\mathbb{T})$[3] is its compact subspace. We further show that the automorphisms group $\mathrm{Aut}(\mathbb{Q},<)$ acts continuously on $\hom(\mathcal{I},\mathbb{T})$, and we use a theorem by Pestov to conclude that $\mathcal{I}$ has a solution if and only if it has a monotone-equivariant solution.

Theorem 10 (Pestov [43])**.** *If the topological group $\mathrm{Aut}(\mathbb{Q},<)$ acts continuously on a nonempty compact space then this action has a fixpoint.*

[3]Recall that a solution can be seen as a homomorphism.

It follows from general principles of equivariant computation on orbit-finite structures [10, 7] that the existence of a monotone-equivariant solution can be decided, which concludes the proof.

In [P1] we also show that many results regarding the complexity, as well as the algebraic properties, of the classical constraint satisfaction problem can be generalized to the case of definable, locally finite templates. An example is Theorem 3, i.e., the characterisation of templates of bounded width. A locally finite template $\mathbb{T}$ has bounded width if there exist numbers $k \leq l$ such that for every finite instance over $\mathbb{T}$ the (k, l)-consistency algorithm provides the correct anwser. Note that this definition is exactly the same as for finite templates, most importantly it concerns only *finite* instances.

Theorem 11. *Let $\mathbb{T}$ be a locally finite structure that is a core. Then $\mathbb{T}$ has bounded width if and only if $\mathbb{T}$ has a pair of weak near unanimity polymorphisms v and w of arity 3 and 4, respectively, satisfying $v(y, x, x) \approx w(y, x, x, x)$.*

A relational structure is a *core* if every unary polymorphism of $\mathbb{T}$ is a monomorphism (this extends the notion of a core for finite structures). We show that for a definable, locally finite template $\mathbb{T}$ a computationally equivalent core can be effectively constructed.

To prove Theorem 11 we first make an easy observation that $\mathbb{T}$ has bounded width if and only if its every finite *subtemplate* has bounded width, where by a *subtemplate* we mean a relational structure whose domain is a subset of the domain of $\mathbb{T}$, and whose relations are relations from $\mathbb{T}$. We say that an identity $s \approx t$ is *linear* if s and t are terms with at most one function symbol. We use a compactness argument to prove that a locally finite template has polymorphisms that satisfy a set of linear identities if and only if its every finite subtemplate has this property. This, together with the characterisation of bounded width for finite templates (see Theorem 3), finishes the proof.

We also show that if the template $\mathbb{T}$ is definable then the condition given in Theorem 11 is effective. We recall the standard construction (see e.g. [2]) that allows us to view polymorphisms of $\mathbb{T}$ satisfying a set of identities as a solution to a certain CSP instance over $\mathbb{T}$. We take the instance whose solutions are pairs of polymorphisms v and w characterising bounded width, and notice that it is definable and computable from $\mathbb{T}$. Hence, by Theorem 9 the existence of such polymorphisms can be decided.

We further study in more detail the complexity of deciding definable instances over finite templates. For a computational class $\mathcal{L}$ we define an "exponentially

bigger" class $\exp(\mathcal{L})$. The formal definition uses padding; here let us only mention that $\exp(\text{L}) = \text{PSpace}$, $\exp(\text{PTime}) = \text{ExpTime}$, and so on. Recall that by $\mathrm{CSP}(\mathbb{T})$ we denote the classical constraint satisfaction problem whose instances are finite CSP instances over a fixed template $\mathbb{T}$. We obtain the following

Theorem 12. *Let $\mathbb{T}$ be a finite template such that $\mathrm{CSP}(\mathbb{T})$ is complete for a complexity class $\mathcal{L}$ under logarithmic space reductions. Then deciding definable instances over $\mathbb{T}$ is complete for the complexity class $\exp(\mathcal{L})$ under logarithmic space reductions.*

To decide whether an order-definable instance $\mathcal{I}$ over a finite template $\mathbb{T}$ has a monotone-equivariant solution we construct a finite instance $\mathcal{I}^*$ over $\mathbb{T}$, whose variables are orbits of $\mathcal{I}$ (with respect to the action of the group $\mathrm{Aut}(\mathbb{Q}, <)$). It has a solution if and only if $\mathcal{I}$ has a monotone-equivariant solution. If $\mathcal{I}$ is definable over $\mathcal{A}toms$ that are natural numbers with equality, then the size of $\mathcal{I}^*$ is at most exponential in the size of the set expression defining $\mathcal{I}$. This results in an exponential blow-up in complexity, and proves the upper bound in Theorem 12. To show the lower bound we use a padding argument.

Theorem 12 implies, for example, that 3-colorability of definable graphs is NExpTime-complete.

3.2. When do Turing machines with atoms determinise? [P2]

In [P2] we fix $\mathcal{A}toms$ to be natural numbers with the equality relation, and we consider Turing machines with atoms (TMA) that are defined almost the same as the classical Turing machines, only their input and work alphabets, state space, and transition relation are assumed to be orbit-finite sets with atoms.

Example 17. Let the input alphabet be the set of atoms. Consider a Turing machine with a state space $\mathcal{A}toms \cup \{\bot, \mathit{accept}\}$, where $\bot$ is the initial state. The machine reads the first letter a of the input word and loads it into the state. Then it scans the word and accepts if it finds another letter a. This machine recognises the language of those words where the first letter appears at least twice (the machine is in fact a deterministic automaton).

In [9] it was shown that there exists an orbit-finite alphabet A, and a language L over A recognised by some nondeterministic TMA, such that no deterministic TMA recognises L. This motivated a definition of *standard* alphabets, i.e., alphabets for which the set of languages recognised by deterministic TMAs

is equal to the set of languages recognised by nondeterministic TMAs. It was left open whether being standard is a decidable property of alphabets. In [P2] we use the constraint satisfaction framework to answer this question in the positive.

In this section, whenever we talk about a relational structure we assume it to be finite, and its vertices to be atoms (every vertex is a different atom). We present our results in the language of so-called *patched structures*.

Let $\mathcal{P}$ be a finite set of finite relational structures. Then the set $A_{\mathcal{P}}$ of all relational structures isomorphic to some structure in $\mathcal{P}$ is an orbit-finite set with atoms. Moreover, it is not difficult to see that every orbit-finite set with atoms is related by an equivariant bijection to $A_{\mathcal{P}}$, for some finite set of relational structures $\mathcal{P}$. Hence, we can assume all alphabets under consideration to be of this form. The intuition is that the structures in $\mathcal{P}$ specify the "shapes" of letters in $A_{\mathcal{P}}$.

Whole words over an alphabet $A_{\mathcal{P}}$ also correspond to certain structures. For a finite set $\mathcal{P}$ of relational structures, a *linearly $\mathcal{P}$-patched structure* is a relational structure $\mathbb{K}$ together with a linearly ordered family of *patches* that cover $\mathbb{K}$. A *patch* is a substructure (not necessarily induced substructure) of $\mathbb{K}$ isomorphic to some element of $\mathcal{P}$. We assume that every vertex and every hyperedge of $\mathbb{K}$ belongs to some patch.

Example 18. If the family $\mathcal{P}$ contains a single undirected graph with two vertices and an edge between them, then a linearly $\mathcal{P}$-patched structure is an undirected graph without isolated vertices with a linear order on the set of edges.

Observe that there is a one-to-one correspondence between words over the alphabet $A_{\mathcal{P}}$ and linearly $\mathcal{P}$-patched structures. Therefore, TMAs can be seen as recognisers of linearly patched structures.

An isomorphism between linearly $\mathcal{P}$-patched structures is a sequence of isomorphisms between corresponding patches (the i-th patch of the first structure has to be mapped to the i-th patch of the second structure) that forms an isomorphism of the whole structures. The starting point to the characterisation of standard alphabets is the following theorem of Bojańczyk et al.

Theorem 13 (Bojańczyk, Klin, Lasota, Toruńczyk [9])**.** *An alphabet A is standard if and only if the language*

$$\{vw \mid v \text{ and } w \text{ are isomorphic}\}$$

is recognised by a deterministic TMA.

Informally, the alphabet $A_{\mathcal{P}}$ is standard if and only if the isomorphism problem for linearly $\mathcal{P}$-patched structures can be solved by a deterministic TMA.

The initial insight of [P2] is that the isomorphism problem for linearly $\mathcal{P}$-patched structures can be expressed as a CSP instance. For any linearly $\mathcal{P}$-patched structure $\mathbb{M}$ we define an equivalence relation $\sim_{\mathbb{M}}$ on the set of its vertices. The relation $\sim_{\mathbb{M}}$ holds between two vertices if they belong to exactly the same patches. The equivalence classes of this relation, called *bags*, inherit a linear order from the lexicographic order induced by the order of the patches of $\mathbb{M}$. The i-th bag is denoted by $B_i^{\mathbb{M}}$. Now, let $\mathbb{M}$ and $\mathbb{K}$ be linearly $\mathcal{P}$-patched structures. The variables of an instance $\mathcal{I}_{\mathbb{K}}^{\mathbb{M}}$ are pairs of corresponding bags $(B_i^{\mathbb{M}}, B_i^{\mathbb{K}})$, and the values are bijections between them (we only do the construction for so-called "similar" structures, where the corresponding bags have the same size and belong to the same patches). For every pair of corresponding patches $\mathfrak{p}_j^{\mathbb{M}}$ and $\mathfrak{p}_j^{\mathbb{K}}$ of structures $\mathbb{M}$ and $\mathbb{K}$, respectively, there is a constraint which says that the bijections from the bags contained in the patch $\mathfrak{p}_j^{\mathbb{M}}$ to the bags contained in the patch $\mathfrak{p}_j^{\mathbb{K}}$ form an isomorphism from $\mathfrak{p}_j^{\mathbb{M}}$ to $\mathfrak{p}_j^{\mathbb{K}}$.

We show that the isomorphism problem from Theorem 13 is recognisable by a deterministic TMA if and only if there exist numbers k and l such that the (k, l)-consistency algorithm provides a correct answer for every instance $\mathcal{I}_{\mathbb{K}}^{\mathbb{M}}$. The right-to-left implication follows from the fact that a deterministic TMA can perform the consistency algorithm. For the other direction, we need to show that if $A_{\mathcal{P}}$ is standard, and the consistency algorithm answers 'YES' then the input structures are indeed isomorphic. This we prove by showing that the nonempty family of partial solutions constructed by the algorithm can be used to translate an accepting run of a TMA over a word vv, to an accepting run over a word vw (the parameters k, l are chosen depending on the Turing machine that solves the isomorphism problem).

For a finite set of structures $\mathcal{P}$, all CSP instances $\mathcal{I}_{\mathbb{K}}^{\mathbb{M}}$ are over some orbit-finite template $\mathbb{T}_{\mathcal{P}}$. Moreover, it can be shown that any instance of $\mathrm{CSP}(\mathbb{T}_{\mathcal{P}})$ is essentially of the form $\mathcal{I}_{\mathbb{K}}^{\mathbb{M}}$, for some structures $\mathbb{M}$ and $\mathbb{K}$. The above result can be therefore phrased as follows:

Theorem 14. *For any finite set of relational structures $\mathcal{P}$, the alphabet $A_{\mathcal{P}}$ is standard if and only if the template $\mathbb{T}_{\mathcal{P}}$ has bounded width.*

The final step is to show that the above characterisation of standard alphabets is effective. In [P2] we do this by constructing a finite structure that has bounded width if and only if $\mathbb{T}_{\mathcal{P}}$ has bounded width. The construction is natural but technically involved. In [P1] we were able to significantly simplify this argument.

Since $\mathbb{T}_{\mathcal{P}}$ is an orbit-finite, locally finite template computable from $\mathcal{P}$, we use Theorem 11 to conclude the decidability of bounded width.

In the last part of [P2] we relate our results concerning TMAs to logics over relational structures. The least fixpoint logic (LFP) is an extension of first order logic by a fixpoint operator (for a precise definition see e.g. [27]). Formulas of LFP can be evaluated over linearly $\mathcal{P}$-patched structures. It turns out that the expressive power of this logic is the same as the expressive power of deterministic TMAs, i.e., for a set of linearly $\mathcal{P}$-patched structures $\mathcal{M}$, we show that the following conditions are equivalent:

- there exists an LFP formula ϕ such that $\mathcal{M} = \{\mathbb{M} \mid \mathbb{M} \models \phi\}$,
- there exists a deterministic TMA that accepts exactly the structures in $\mathcal{M}$.

By adding to LFP the possibility of counting we obtain the least fixpoint logic with counting (LFP+C) (see e.g. [27]). The above result holds also for LFP+C, since the expressive power of those logics over linearly $\mathcal{P}$-patched structures is the same.

The main open problem in descriptive complexity theory asks about the existence of *logic for PTime*. By this we mean a logic whose formulas would define exactly the same sets of structures as sets of structures whose encodings are recognised by (classical) deterministic Turing machines in polynomial time (for details see e.g. [27]). A famous result independently showed by Immerman and Vardi [32, 48] says that LFP captures PTime over ordered strucutres (structures with a linear order on the set of vertices). On the other hand, Cai, Fürer, and Immerman [17] proved that even the more expressive logic LFP+C does not capture PTime over all structures. Using our results on TMAs we obtain a common generalisation of those results.

Theorem 15. *For a finite set of relational structures $\mathcal{P}$, the logic LFP (and LFP+C) captures polynomial time over linearly $\mathcal{P}$-patched structures if and only if the alphabet $A_{\mathcal{P}}$ is standard.*

In the light of Theorems 11 and 14 the above characterisation is effective.

3.3. A necessary condition for tractability of valued CSPs [P3]

In [P3] we no longer work within the framework of sets with atoms. We consider a different generalisation of the classical CSP, namely, the valued constraint satisfaction problem (VCSP), which fall into the area of discrete optimisation.

Before presenting the contribution of [P3] we provide basic definitions and examples showing how the main CSP notions are modified to capture the optimisation aspect. For details we refer to the survey by Jeavons et al. [34].

By $\overline{\mathbb{Q}} = \mathbb{Q} \cup \{\infty\}$ we denote the set of rational numbers with (positive) infinity. A *cost function* on a set D of arity n is a function $\varrho\colon D^n \to \overline{\mathbb{Q}}$. A *feasibility relation* of ϱ is defined by $\mathrm{Feas}(\varrho) = \{\bar{x} \in D^n \mid \varrho(\bar{x}) \text{ is finite}\}$. Whenever we talk about polymorphisms of cost functions we mean polymorphisms (in the sense of Section 1) of their feasibility relations.

In an instance of the valued constraint satisfaction problem the constraints not only specify the allowed combinations of values, but also a cost of each such combination. The goal is to minimise an aggregate cost of an assignment.

Definition 6 (VCSP Instance)**.** An *instance* of the valued constraint satisfaction problem is a triple $(V, D, \mathcal{C})$, where V is a set of *variables*, D is a set of their possible *values*, called a *domain*, and $\mathcal{C}$ is a set of *constraints*. A *constraint* is a pair $(\bar{x}, \varrho)$, where $\bar{x}$ is an n-tuple of variables, and $\varrho\colon D^n \to \overline{\mathbb{Q}}$ is an n-ary cost function on D.

The *cost* of an assignment $f\colon V \to D$ is given by $Cost(f) = \sum_{(\bar{x},\varrho)\in\mathcal{C}} \varrho(f(\bar{x}))$ (where f is applied component-wise). To *solve* $\mathcal{I}$ is to find an assignment with a minimal cost, called an *optimal assignment*.

Many natural optimisation problems can be phrased in the valued constraint satisfaction framework.

Example 19 (Max-Cut)**.** In the Max-Cut problem, one needs to find a partition of the vertices of a given graph into two sets, such that the number of edges with ends in different sets is maximal.

The Max-Cut problem can be expressed as a valued constraint satisfaction problem. The domain has two elements 0 and 1. Variables in the instance are vertices of the graph, and for each edge e there is a constraint of a form (e, ϱ_{XOR}), where ϱ_{XOR} is a binary cost function defined by

$$\varrho_{XOR}(x, y) = \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{otherwise.} \end{cases}$$

Any assignment of the values 0 and 1 to the variables corresponds to a partition of the graph. The cost of an assignment is equal to the number of edges of the graph minus the number of cut edges.

Example 20 (Min-Vertex-Cover)**.** A cover of an undirected graph $G(V, E)$ is a subset V' of the set of vertices of the graph, such that every edge is incident to at least one vertex in V'. In the Min-Vertex-Cover problem, one needs to find a cover of a given graph with a minimal number of vertices.

Min-Vertex-Cover seen as a VCSP has a domain $\{0, 1\}$, where assigning 1 to a vertex means choosing it to be in the cover. The set of variables is V. For every edge there is a constraint of the form (e, ϱ_{MVC}), where

$$\varrho_{MVC}(x, y) = \begin{cases} \infty & \text{if } x = y = 0, \\ 0 & \text{otherwise.} \end{cases}$$

Constraints of this kind are used to enforce an assignment of 0's and 1's to the vertices of the graph to be a cover. Moreover, for every vertex there is a constraint of the form (v, ϱ_1), where

$$\varrho_1(x) = \begin{cases} 1 & \text{if } x = 1, \\ 0 & \text{if } x = 0. \end{cases}$$

Those constraints "count" the number of vertices in the cover. If the cost of an assignment is finite then it is equal to the number of vertices in a corresponding cover. If it is infinite then the assignment does not specify a cover.

The valued constraint satisfaction problem generalises the constraint satisfaction problem (as presented in Section 1). CSP instances correspond to those VCSP instances where all cost functions in all constraints assign only cost 0 or ∞ to every tuple. This is because such cost functions can be seen as relations.

In [P3] we consider non-uniform VCSPs. That is, for a finite set of cost functions Γ (over a fixed domain D) called a *valued constraint language*, we consider the problem $\mathrm{VCSP}(\Gamma)$ whose instances are VCSP instances where all cost functions in all constraints come from Γ.

We study the complexity of such problems using the algebraic approach introduced in [20] (which is a generalisation of the algebraic approach to the CSP). The key notion is that of a *weighted polymorphism*. Let $\mathrm{Pol}_k(\Gamma)$ denote the set of all k-ary polymorphisms of a valued constraint language Γ.

Definition 7 (Weighted polymorphism)**.** Let Γ be a valued constraint language. A function $\omega\colon \mathrm{Pol}_k(\Gamma) \to \mathbb{Q}$ is a *weighted polymorphism* of Γ if it satisfies the following conditions:

- $\sum_{f \in \mathrm{Pol}_k(\Gamma)} \omega(f) = 0$,
- if $\omega(f) < 0$ then f is a projection,
- for any cost function $\varrho \in \Gamma$, and any list of n-tuples $\bar{x}_1, \ldots, \bar{x}_k \in \mathrm{Feas}(\varrho)$,

$$\sum_{f \in \mathrm{Pol}_k(\Gamma)} \omega(f) \cdot \varrho(f(\bar{x}_1, \ldots, \bar{x}_k)) \leq 0, \text{ where } f \text{ is applied component-wise.}$$

Weighted polymorphisms characterise the complexity of VCSPs [20] in a similar manner as polymorphisms characterise the complexity of CSPs.

In [P3] we introduce and study a notion of a *positive clone* $\mathrm{Pol}^+(\Gamma)$ of a constraint language Γ. A positive clone contains all those polymorphism of Γ which are assigned a positive weight by some weighted polymorphism of Γ. Our key result generalises Theorem 2:

Theorem 16. *Let Γ be a valued constraint language that is a core. If there is no idempotent cyclic polymorphism in $\mathrm{Pol}^+(\Gamma)$, then $\mathrm{VCSP}(\Gamma)$ is NP-hard.*

A notion of a *core language* extends the notion of a core template. A valued constraint language Γ is a core if all unary polymorphisms in $\mathrm{Pol}^+(\Gamma)$ are bijective. We show that every valued constraint language has a computationally equivalent core language (using similar arguments as in the previously known result for the special case of so-called *finite-valued* CSPs [31, 47]).

The key idea of the proof of Theorem 16 lies in the fact that any relation compatible with $\mathrm{Pol}^+(\Gamma)$ can be encoded as an instance of $\mathrm{VCSP}(\Gamma)$. This we show using Farkas' lemma. If there is no idempotent cyclic polymorphism in $\mathrm{Pol}^+(\Gamma)$ then there is a relation R that corresponds to a certain NP-hard problem and is compatible with $\mathrm{Pol}^+(\Gamma)$ [46, 3]. This ends the proof.

We further conjecture that the lack of a cyclic polymorphism in the positive clone is the only reason for intractability of VCSPs (this generalises the Algebraic Dichotomy Conjecture).

Conjecture 3 (VCSP Algebraic Dichotomy Conjecture)**.** *Let Γ be a valued constraint language that is a core. If there is no idempotent cyclic polymorphism in $\mathrm{Pol}^+(\Gamma)$, then $\mathrm{VCSP}(\Gamma)$ is NP-hard. Otherwise it is polynomial-time solvable.*

Finally, we show that the above conjecture agrees with all known complexity classifications for valued constraint satisfaction problems.

It is worth mentioning that shortly after [P3] was published Kolmogorov et al. [36] proved a result which together with Theorem 16 implies an equivalence of the CSP and VCSP dichotomy conjectures. Specifically, they showed that for a core language Γ, if the underlying feasibility relations form a tractable CSP language, and there is an idempotent cyclic polymorphism in $\mathrm{Pol}^{+}(\Gamma)$, then $\mathrm{VCSP}(\Gamma)$ is tractable.

Bibliography

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] Libor Barto. The collapse of the bounded width hierarchy. *Journal of Logic and Computation*, 2014.

[3] Libor Barto and Marcin Kozik. Absorbing subalgebras, cyclic terms, and the constraint satisfaction problem. *Logical Methods in Computer Science*, 8(1), 2012.

[4] Libor Barto and Marcin Kozik. Constraint satisfaction problems solvable by local consistency methods. *J. ACM*, 61(1):3:1–3:19, January 2014.

[5] George Birkhoff. On the structure of abstract algebras. In *Proceedings of the Cambridge Philosophical Society*, volume 31, pages 433–454, 1935.

[6] Mikołaj Bojańczyk. Computation in sets with atoms. Manuscript. *http://atoms.mimuw.edu.pl*, November 2013.

[7] Mikołaj Bojańczyk, Laurent Braud, Bartek Klin, and Sławomir Lasota. Towards nominal computation. In *Proc. POPL'12*, pages 401–412, New York, 2012. ACM.

[8] Mikołaj Bojańczyk, Bartek Klin, and Sławomir Lasota. Automata theory in nominal sets. *Logical Methods in Computer Science*, 10, 2014.

[9] Mikołaj Bojańczyk, Bartek Klin, Sławomir Lasota, and Szymon Toruńczyk. Turing machines with atoms. In *Proc. LICS'13*, pages 183–192, 2013.

[10] Mikołaj Bojańczyk and Szymon Toruńczyk. Imperative programming in sets with atoms. In *Proc. FSTTCS 2012*, volume 18 of *LIPIcs*, pages 4–15, 2012.

[11] Andrei A. Bulatov. Tractable conservative constraint satisfaction problems. In *Proc. LICS'03*, pages 321–330, 2003.

[12] Andrei A. Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *J. ACM*, 53(1):66–120, January 2006.

[13] Andrei A. Bulatov. Bounded relational width. Manuscript, 2009.

[14] Andrei A. Bulatov, Peter Jeavons, and Andrei A. Krokhin. Classifying the complexity of constraints using finite algebras. *SIAM J. Comput.*, 34:720–742, 2005.

[15] Andrei A. Bulatov, Andrei A. Krokhin, and Peter Jeavons. Constraint satisfaction problems and finite algebras. In *Proc. ICALP '00*, pages 272–282, 2000.

[16] Stanley Burris and H. P. Sankappanavar. *A Course in Universal Algebra*. Number 78 in Graduate Texts in Mathematics. Springer-Verlag, 1981.

[17] Jin-Yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992.

[18] Vincenzo Ciancia. Accessible functors and final coalgebras for named sets. PhD thesis, University of Pisa, 2008.

[19] Vincenzo Ciancia and Ugo Montanari. Symmetries, local names and dynamic (de)-allocation of names. *Information and Computation*, 208(12):1349 – 1367, 2010.

[20] David Cohen, Martin Cooper, Páidí Creed, Peter Jeavons, and Stanislav Živný. An algebraic theory of complexity for discrete optimization. *SIAM J. Comput.*, 42(5):1915–1939, 2013.

[21] Vìctor Dalmau, Phokion G. Kolaitis, and Moshe Y. Vardi. Constraint satisfaction, bounded treewidth, and finite-variable logics. In *Principles and Practice of Constraint Programming*, volume 2470 of *LNCS*, pages 310–326. Springer Berlin Heidelberg, 2002.

[22] Rodney Downey and Michael Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer-Verlag New York, 1999.

[23] Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory. *SIAM J. Comput.*, 28(1):57–104, February 1999.

[24] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. An EATCS Series. Springer-Verlag Berlin Heidelberg, 2006.

[25] Murdoch J. Gabbay and Andrew M. Pitts. A new approach to abstract syntax with variable binding. *Formal Aspects of Computing*, 13(3-5):341–363, 2002.

[26] Martin Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *J. ACM*, 54(1):1:1–1:24, March 2007.

[27] Martin Grohe. Descriptive complexity, canonisation, and definable graph structure theory. *http://www.automata.rwth-aachen.de/~grohe/cap/index.en*, December 2013.

[28] David Hobby and Ralph McKenzie. *The structure of finite algebras*, volume 76 of *Contemporary Mathematics*. American Mathematical Society, 1988.

[29] Wilfrid Hodges. *A shorter model theory*. Cambridge University Press, Cambridge, 1997.

[30] Karel Hrbacek and Thomas Jech. *Introduction to set theory*. Monographs and textbooks in pure and applied mathematics. M. Dekker, New York, 1999.

[31] Anna Huber, Andrei A. Krokhin, and Robert Powell. Skew bisubmodularity and valued CSPs. In *Proc. SODA '13*, pages 1296–1305. SIAM, 2013.

[32] Neil Immerman. Upper and lower bounds for first order expressibility. *Journal of Computer and System Sciences*, 25(1):76 – 98, 1982.

[33] Peter Jeavons. On the algebraic structure of combinatorial problems. *Theoretical Computer Science*, 200(1-2):185 – 204, 1998.

[34] Peter Jeavons, Andrei A. Krokhin, and Stanislav Živný. The complexity of valued constraint satisfaction. *Bulletin of the EATCS*, 113, 2014.

[35] John Kelley. *General topology*. Springer-Verlag New York, 1975.

[36] Vladimir Kolmogorov, Andrei A. Krokhin, and Michal Rolinek. The complexity of general-valued CSPs. *CoRR*, abs/1502.07327, 2015.

[37] Marcin Kozik, Andrei A. Krokhin, Matt Valeriote, and Ross Willard. Characterizations of several maltsev conditions. *Algebra universalis*, 73(3-4):205–224, 2015.

[38] Richard Ladner. On the structure of polynomial time reducibility. *J. ACM*, 22(1):155–171, January 1975.

[39] Benoit Larose and Pascal Tesson. Universal algebra and hardness results for constraint satisfaction problems. *Theoretical Computer Science*, 410(18):1629 – 1647, 2009.

[40] Benoit Larose, Matthew Valeriote, and László Zádori. Omitting types, bounded width and the ability to count. *IJAC*, 19(5):647–668, 2009.

[41] Benoit Larose and László Zádori. Bounded width problems and algebras. *Algebra universalis*, 56(3-4):439–466, 2007.

[42] Miklós Maróti and Ralph McKenzie. Existence theorems for weakly symmetric operations. *Algebra universalis*, 59(3-4):463–489, 2008.

[43] Vladimir Pestov. On free actions, minimal flows, and a problem by ellis. *Trans. of the American Mathematical Society*, 350(10):pp. 4149–4165, 1998.

[44] Andrew M. Pitts. *Nominal Sets: Names and Symmetry in Computer Science*. Cambridge University Press, New York, 2013.

[45] Thomas Schaefer. The complexity of satisfiability problems. In *Proc. of the 10th ACM Symp. on Theory of Computing*, STOC '78, pages 216–226, 1978.

[46] Walter Taylor. Varieties obeying homotopy laws. *Canadian Journal of Mathematics*, 29(3):498–527, 1997.

[47] Johan Thapper and Stanislav Živný. The complexity of finite-valued CSPs. In *Proc. of the 45th ACM Symp. on Theory of Computing*, STOC '13, pages 695–704, 2013.

[48] Moshe Y. Vardi. The complexity of relational query languages (extended abstract). In *Proc. of the 14th ACM Symp. on Theory of Computing*, STOC '82, pages 137–146, 1982.

Locally Finite Constraint Satisfaction Problems

Bartek Klin*, Eryk Kopczyński†, Joanna Ochremiak*, Szymon Toruńczyk*
University of Warsaw
Email: {klin,erykk,ochremiak,szymtor}@mimuw.edu.pl

***Abstract*—First-order definable structures with atoms are infinite, but exhibit enough symmetry to be effectively manipulated. We study Constraint Satisfaction Problems (CSPs) where both the instance and the template are definable structures with atoms. As an initial step, we consider locally finite templates, which contain potentially infinitely many finite relations. We argue that such templates occur naturally in Descriptive Complexity Theory.**

We study CSPs over such templates for both finite and infinite, definable instances. In the latter case even decidability is not obvious, and to prove it we apply results from topological dynamics. For finite instances, we show that some central results from the classical algebraic theory of CSPs still hold: the complexity is determined by polymorphisms of the template, and the existence of certain polymorphisms, such as majority or Maltsev polymorphisms, guarantees the correctness of classical algorithms for solving finite CSP instances.

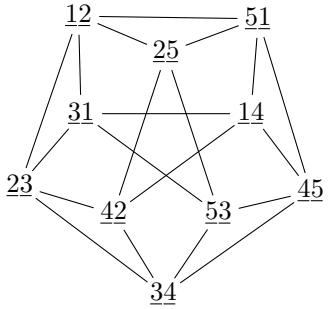
***Index Terms*—Sets with atoms; Constraint Satisfaction Problems**

Once and for all, fix a countably infinite set $\mathcal{A} = \{\underline{1}, \underline{2}, \underline{3}, \ldots\}$, whose elements we call *atoms*.

I. Introduction

Example 1. Here is an easy puzzle: consider an (infinite) graph G with ordered pairs of distinct atoms as vertices (here we denote such a pair simply by ab, for $a \neq b \in \mathcal{A}$), and with an undirected edge ab—bc whenever a and c are distinct. Is this graph 3-colorable?

The answer is negative, as G contains the subgraph:



checked by hand, is not 3-colorable. We do not know whether this is the smallest non-3-colorable subgraph of G, but we could not find a smaller one, and it is rather interesting to see how big a graph we needed to check.

This motivates a harder puzzle: consider graphs with n-tuples of distinct atoms as vertices, and edges defined by quantifier-free (or even first order) formulas with equality, and with $2n$ variables that range over atoms; for G above, $n = 2$ and the set of edges is:

$$\big\{\{ab, dc\} \mid a, b, c, d \in \mathcal{A}, (b = d) \wedge (a \neq b) \wedge (c \neq d) \wedge (a \neq c)\big\}.$$

Now the question is whether the 3-colorability of a graph represented by a number n and a formula is decidable at all? It is a standard exercise in logical compactness that a graph is 3-colorable iff all its finite subgraphs are, but it may not be clear whether there is a computable bound on the size of finite subgraphs that need to be checked to ensure the colorability of the entire graph. □

Example 2. Systems of linear equations over the two-element field $\mathbb{Z}_2$ can be augmented with atoms just as graphs can. Consider n-tuples of distinct atoms as variable names, and let a system of equations be defined by a formula similarly to Example 1, for example (with $n = 2$):

$$\big\{ab + bc + ca = 0 \mid a, b, c \in \mathcal{A}, (a \neq b) \wedge (b \neq c) \wedge (a \neq c)\big\}.$$

This system has a trivial solution where all variables have value 0. To disallow that solution one may e.g. extend the system by one more equation:

$$\underline{12} + \underline{21} = 1.$$

Does the extended system have a solution? It turns out that it does not, but a finite subsystem with no solutions again turns out curiously bulky. Here is the smallest one that we have managed to find:

$$\begin{array}{cccccccccccccc}
\underline{12} & + & \underline{21} & & & & & & & & & & & = 1 \\
\underline{12} & & + & \underline{23} & + & \underline{31} & & & & & & & & = 0 \\
& \underline{21} & & & + & \underline{13} & + & \underline{32} & & & & & & = 0 \\
& & \underline{23} & & & & + & \underline{34} & + & \underline{42} & & & & = 0 \\
& & & \underline{31} & & & & & + & \underline{15} & & + & \underline{53} & = 0 \\
& & & & \underline{13} & & + & \underline{34} & & & & & + & \underline{41} = 0 \\
& & & & & \underline{32} & & & & & + & \underline{25} & + & \underline{53} = 0 \\
& & & & & & & & \underline{42} & & + & \underline{25} & & + & \underline{54} = 0 \\
& & & & & & & & & \underline{15} & & & + & \underline{54} + \underline{41} = 0 \\
\hline
0 & + & 0 & + & 0 & + & 0 & + & 0 & + & 0 & + & 0 & = 1
\end{array}$$

Again, a question appears whether the solvability of equation systems given by formulas over atoms is decidable, and if so, what its complexity is. □

The above are examples of so-called *Constraint Satisfaction Problems* (CSPs). An instance $\mathbb{I}$ of a CSP is a set of *variables* together with a set of *constraints* of the form

Supported by the Polish National Science Centre (NCN) grants *2012/07/B/ST6/01497 and †2012/07/D/ST6/02435.

$\big((x_1,\dots,x_n),R\big)$, where the x_i are variables and R is an n-ary relation belonging to a fixed family of relations $\mathcal{R}$ over a *domain* T; the pair $\mathbb{T}=(T,\mathcal{R})$ is called a *template* for $\mathbb{I}$.

For example, 3-colorability is a CSP for a template with three elements (colors) equipped with a single binary inequality relation $\neq$. To see a graph as an instance, one considers its vertices as variables, and adds a constraint $\big((x,y),\neq\big)$ whenever x and y are adjacent. An equation system $\mathbb{E}$ over $\mathbb{Z}_2$, assuming that every equation is of the form $x+y+z=0$ or $x+y=1$, can be seen as an instance over a template with two elements 0 and 1, equipped with two relations:

$$\begin{array}{rcl} Z & = & \{(0,0,0),(0,1,1),(1,0,1),(1,1,0)\},\\ S & = & \{(0,1),(1,0)\}. \end{array}$$

To construct an instance, one picks constraints:

$$\begin{array}{rl} \big((x,y,z),Z\big) & \text{for each} \quad x+y+z=0 \text{ in } \mathbb{E},\\ \big((x,y),S\big) & \text{for each} \quad x+y=1 \text{ in } \mathbb{E}. \end{array}$$

A *solution* of an instance is an assignment f which maps every variable to a template element, so that for every constraint $\big((x_1,\dots,x_n),R\big)$, the tuple of values $\big(f(x_1,),\dots,f(x_n)\big)$ belongs to the relation R. It is useful to view a template $\mathbb{T}=(T,\mathcal{R})$ as a relational structure with universe[1] T, over the signature $\mathcal{R}$, with the tautological interpretation mapping. A CSP instance $\mathbb{I}$ over the template $\mathbb{T}$ can then be viewed as a relational structure, whose universe consists of its variables I, and the interpretation of a relation $R\in\mathcal{R}$ of arity n is the set of those tuples $\bar{x}\in I^n$ for which $(\bar{x},R)$ is a constraint. Then solutions of $\mathbb{I}$ correspond to homomorphisms of relational structures from $\mathbb{I}$ to $\mathbb{T}$.

The classical theory of CSPs tries to classify the computational complexity of the following decision problem, parametrized by a template $\mathbb{T}$, with *finite* instances.

Problem: CSP($\mathbb{T}$)
Input: A finite instance $\mathbb{I}$ over $\mathbb{T}$
Decide: Does $\mathbb{I}$ have a solution?

The *algebraic approach* to this end is particularly successful. It is based on the observation that the complexity of CSP($\mathbb{T}$) entirely depends on the algebra of *polymorphisms* (a multivariate generalization of the notion of an endomorphism) of the template $\mathbb{T}$ [1]. For example, the fact that finite systems of equations over $\mathbb{Z}_2$ can be solved in polynomial time can be inferred from the fact that the relevant template has a so-called Maltsev polymorphism [2], and the NP-completeness of graph 3-coloring follows from the fact that the corresponding template has no so-called cyclic polymorphisms [3]–[5].

Our Examples 1 and 2 do not fit the mainstream development of CSP theory, since our instances are infinite. They are, however, definable via first order expressions, in a sense made precise in Section II. The aim of this paper is to formulate the rudiments of CSP theory for definable structures. We define and study the complexity of the following decision problem.

[1]In this paper, we adapt the convention that the universe of a relational structure $\mathbb{A}$ is denoted with the corresponding italic letter A.

Problem: CSP-Inf($\mathbb{T}$)
Input: An expression defining an instance $\mathbb{I}$ over $\mathbb{T}$
Decide: Does $\mathbb{I}$ have a solution?

We show that, for any fixed finite template $\mathbb{T}$, this problem is decidable, and specify tight complexity bounds. In particular, the following result is a consequence of the results proved in Section III.

Theorem 3. *Let $\mathbb{T}$ be a finite template such that* CSP($\mathbb{T}$) *is complete for a complexity class $\mathcal{C}$ under logarithmic space reductions. Then* CSP-Inf($\mathbb{T}$) *is decidable and complete for the complexity class* $\exp(\mathcal{C})$ *under logarithmic space reductions.*

The general definition of the class $\exp(\mathcal{C})$ is given in Section III-B; here we just mention that $\exp(\mathrm{L}) = \textsc{PSpace}, \exp(\mathrm{P}) = \textsc{Exp}, \exp(\mathrm{NP}) = \textsc{NExp}$, etc.

Interestingly, our key technical tool for proving the upper bound comes from topological dynamics, in the following theorem due to Pestov:

Theorem 4 ([6])**.** *Every continuous action of the topological group* $\mathrm{Aut}(\mathbb{Q},\leq)$ *on a compact space has a fixpoint.*

This theorem is strongly related with Ramsey's theorem (see [7] for a generalization of this theorem, linking it to Ramsey theory). In fact, the upper bound in Theorem 3 could be proved directly with the use of Ramsey's theorem.

In Section IV, we reverse the situation and consider finite instances over infinite templates. We allow the templates to have an infinite set of relations, but we assume them to be *locally finite*, i.e., every relation is finite. Examples of such CSPs appear in various contexts:

Example 5. Consider the following *graph coloring* problem. Fix a positive integer k. Let $G=(V,E)$ be a finite graph, together with a labeling $l: V\to\binom{\mathcal{A}}{k}$, where elements of $\mathcal{A}$ are interpreted as *colors*.

The problem is to decide whether one can find a coloring $c: V\to\mathcal{A}$ such that $c(v)\in l(v)$ for every $v\in V$ and $c(v)\neq c(w)$ whenever v and w are adjacent in G.

This problem can be understood as a CSP over a (definable) template whose domain is $\mathcal{A}$, with a relation

$$R_{C,D}=C\times D-\Delta$$

for every pair $C,D\in\binom{\mathcal{A}}{k}$, where $\Delta\subseteq\mathcal{A}^2$ is the binary diagonal relation. The instance corresponding to a graph has V as the set of variables, and a constraint $\big((v,w),R_{l(v),l(w)}\big)$ for each pair of adjacent vertices v,w. □

Note that while every finite instance over an infinite, locally finite template is trivially an instance over its finite subtemplate, there may be no single finite subtemplate that immediately fits all instances of interest. Since templates in CSP theory are means of grouping large classes of similar instances, it may sometimes be useful to consider infinite, locally finite templates.

Situations where the set of admissible values for variables is not fixed for all instances sometimes arise naturally. For

example, consider the *cycle cover problem*: given a directed graph G, decide whether it contains a set of directed cycles so that every vertex belongs to exactly one cycle. This is equivalent to checking whether one can choose an outgoing edge from every vertex so that no two chosen edges have the same target. In other words, one wants to color every vertex with one of its out-neighbours so that no two vertices get the same color. Assuming a bound k on the out-degree of the vertices of G, and labeling vertices of G with atoms in an arbitrary way, this can be seen as a CSP instance over the template from Example 5. Here graph vertices play the role of colors, therefore the set of possible colors depends on the instance.

Another example is that of Cai-Fürer-Immerman (CFI) graphs [8] considered in Descriptive Complexity Theory:

Example 6. Consider a template with atoms as elements and, for every triple of pairs of distinct atoms $\beta = ((a,a'),(b,b'),(c,c'))$, a ternary relation:

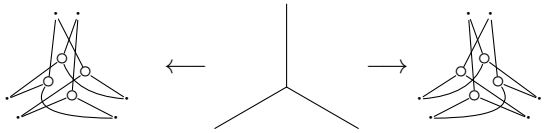
$$R_\beta = \big\{(a,b,c),(a,b',c'),(a',b,c'),(a',b',c)\big\}.$$

To construct an interesting instance over this template, start with a 3-regular graph G and label each edge e with a set $\{a,a'\} \subseteq \mathcal{A}$ of two distinct atoms, so that labels of distinct edges do not intersect. Let the edges of G be the variables of the instance. For each vertex v adjacent to some edges e_1, e_2 and e_3 add exactly one constraint

$$\big((e_1,e_2,e_3),R_\beta\big),$$

where $\beta = \big((a,a'),(b,b'),(c,c')\big)$ arises from some ordering of the unordered labels $\{a,a'\},\{b,b'\},\{c,c'\}$ of the edges e_1,e_2,e_3. Note that even though there are eight possible orderings, there are only two possible resulting relations R_β.

Graphically, this can be seen as replacing every edge with two nodes, and every vertex with one of two little hypergraphs:



Such an instance has a solution if and only if one can choose one of the two nodes for each edge of G so that for each vertex v of G, the three nodes chosen for the adjacent edges are connected by a hyperedge in the hypergraph for v. Checking whether a given instance has a solution is called a *CFI query*, and is the core of the Cai-Fürer-Immerman theorem from Descriptive Complexity (see Section V for more on this topic). □

In Section IV we lift some central results from classical CSP theory to the setting of locally finite templates. First, we prove that the cornerstone result of the algebraic approach to CSP still holds: the complexity of the CSP problem over a template depends only on the set of polymorphisms over that template. We also show that the important notion of a *core* template has the expected properties, and that a definable template can be effectively converted to its core.

Furthermore, we show that a locally finite template admits a family of polymorphisms defined by a specified set of *linear identities* iff each of its finite subtemplates admit such polymorphisms. This proves a series of results analogous to finite CSP theory: for example, if a locally finite template $\mathbb{T}$ has a majority polymorphism or a Maltsev polymorphism then specific polynomial time algorithms correctly solve finite instances over $\mathbb{T}$. Moreover, by the results of Section III, we can effectively decide whether a given definable, locally finite template $\mathbb{T}$ admits polymorphisms which satisfy a specified set of linear identities. This allows us to prove statements such as the one below, which follows from Corollary 35 and Proposition 31 in Section IV, and characterizes those templates for which the CSP problem can be solved by a certain well-known *bounded width algorithm*.

Theorem 7. *Let $\mathbb{T}$ be a definable, locally finite template. It is decidable whether $\mathbb{T}$ has bounded width.*

Coming back to Example 5, one may notice that not only every particular instance there is over a finite template: in fact, the entire graph coloring problem can be presented as a CSP over a single finite template. To do this, impose a total order on $\mathcal{A}$; this defines a bijection from each $l(v)$ to the set $[k] = \{1,2,\dots,k\}$. One can then consider a template of k elements, with a relation

$$R_{i,j} = [k] \times [k] - \{(i,j)\}$$

for each $i,j \in [k]$, and translate a graph to be colored to an instance over that finite template. In Section IV-C, we show how this can be generalized: for any definable, locally finite template $\mathbb{T}$ there is a finite template $\hat{\mathbb{T}}$, and a polynomial time reduction of instances over $\mathbb{T}$ to equivalent instances over $\hat{\mathbb{T}}$. This shows that CSP($\mathbb{T}$) is not computationally harder then CSP($\hat{\mathbb{T}}$). Moreover, if $\hat{\mathbb{T}}$ admits polymorphisms satisfying some linear identities, then so does $\mathbb{T}$. By the algebraic results proved earlier in Section IV, and by results very recently announced by Barto and Pinsker [9], this says that CSP($\hat{\mathbb{T}}$) is not harder than CSP($\mathbb{T}$).

Finally, in Section V we show how locally finite templates streamline the (previously unpleasantly technical) proof of the main result in [10], a characterization of those *linearly patched structures* over which the least fixpoint logic LFP captures polynomial time computations.

Related work

CSP for certain infinite instances were studied in [11]. All instances there are *periodic*, i.e., invariant under an action of a subgroup of finite index of the automorphism group of the total order of integers. This is similar to our approach, as our definable instances are also invariant under certain groups, in particular the automorphism group of rational numbers. However, the choice of a group makes a big difference: thanks to model-theoretic properties of rationals we are able to prove decidability results that do not hold in the setting of [11]. Indeed, one of the main points of that paper was to show

that for periodic instances, 3-colorability is *un*decidable. Proof techniques used there are also quite different from ours.

The line of research started in [11] was continued in [12], and certain infinite instances were studied also in [13]. In [12], [13] it is argued that infinite periodic instances naturally arise when studying large – perhaps of unknown size or infinite – constraint networks whose constraints possess a high degree of regularity or symmetry. We believe that relaxing the periodicity assumption might be natural in many cases, and, as we show, leads to a drastic improvement in the complexity (in the case of 3-colorability, from undecidable by [11] to EXP by Theorem 3). It would be interesting to look for a common generalisation of these developments and ours.

More attention has been devoted to the study of finite instances over infinite templates. In contrast to our work, the templates there usually consist of only finitely many infinite relations. In the most well-behaved case of *ω-categorical* templates, central results of finite CSP still hold [14]–[16]. In particular, the complexity of templates depend only on their polymorphisms, which gives complexity classifications for large classes of ω-categorical templates [17]–[19]. Connections to Ramsey theory were studied in [20]. Our section IV shows that some of these results hold for locally finite templates as well.

Sets with atoms are also known in Computer Science as nominal sets [21]. In fact, our notion of definable set is almost the same as the notion of orbit-finite set considered there: every definable set is orbit-finite, and every orbit-finite nominal set is isomorphic to a definable one [22]. We choose to define our sets by first order formulas, but all results we show here could be reformulated in terms of group actions, orbit-finite sets and finite supports, studied in [21]. In fact, we used that terminology in most previous work on computation theory over sets with atoms [10], [23], [24], of which the present paper is a natural continuation.

II. SETS WITH ATOMS

A. Definable sets

We introduce definable sets with atoms as follows. An *expression* is either a variable ranging over atoms, or a finite tuple of expressions, or a formal finite union of expressions, or an integer, or a *set-builder expression*, which is a variable binding construct of the form

$$\{\ e \mid v_1, \dots, v_n \in \mathcal{A}, \phi\},$$

where e is an expression, $v_1, \dots, v_n$ are bound variables, and ϕ is a first order formula with equality as the only predicate, whose free variables are contained in the free variables of e. A *quantifier-free* expression is an expression which uses only quantifier-free formulas, on every level, recursively.

If an expression e has free variables V, then any valuation $\mathrm{val} : V \to \mathcal{A}$ defines in an obvious way the value $X = e[\mathrm{val}]$, which is either a set, an atom, a tuple, or an integer.[2] We say that X is a *definable set with atoms* and that it is *defined* by e with valuation val. Note that the same set X can be defined by many different expressions. We denote by $\mathcal{D}$ the set of all definable sets. Observe that any element of a definable set is definable.

Example 8. Examples of definable sets with atoms include:

- any atom, such as $\underline{1}$, as defined by the expression v, with valuation $v \mapsto \underline{1}$.
- any pair of atoms, such as $(\underline{1}, \underline{2})$, as defined by the expression (v, w), with valuation $v \mapsto \underline{1}, w \mapsto \underline{2}$.
- the set $\mathcal{A}$ of atoms itself, as defined by the expression $\{v \mid v \in \mathcal{A}\}$.
- for any $n \in \mathbb{N}$, the set $\mathcal{A}^n$ of all n-tuples and $\mathcal{A}^{(n)}$ of non-repeating n-tuples of atoms, as defined by the expression

  $$\{(v_1, \dots, v_n) \mid v_1, \dots, v_n \in \mathcal{A}, \phi\},$$

  where ϕ is $\top$ in the case of $\mathcal{A}^n$ and $\bigwedge_{1 \le i < j \le n} (v_i \neq v_j)$ in the case of $\mathcal{A}^{(n)}$.

An example of a definable function is the swapping function $s : \mathcal{A}^2 \to \mathcal{A}^2$, $s(a, b) = (b, a)$, whose graph is defined by the expression $\{((v, w), (w, v)) \mid v, w \in \mathcal{A}\}$. □

For any mathematical object (a relation, a function, a logical structure, etc.), it makes sense to ask whether it is definable. E.g., a definable relation on X, Y is a relation $R \subseteq X \times Y$ which is a definable set. As a side remark, definable structures over a finite signature correspond, up to isomorphism, to structures which *interpret* in $\mathcal{A}$ (a notion from logic).

As a particular case of the above definition, a *definable instance* is an instance $I = (V, C)$ such that the set of variables V and the set of constraints C are definable. Definable instances are represented by expressions, which are used as inputs for the problem CSP-Inf($\mathbb{T}$) described in the Introduction.

Example 9. We show an expression describing the instance from Example 1. Consider the following expressions.

$$R : \big\{(0,1),(0,2),(1,0),(1,2),(2,0),(2,1)\big\}$$
$$V : \big\{(a,b) \mid a, b \in \mathcal{A}, a \neq b\big\}$$
$$C : \big\{\big(((a,b),(b,c)), R\big) \mid a, b, c \in \mathcal{A}, a \neq b \wedge b \neq c \wedge a \neq c\big\}$$
$$I : (V, C)$$

They define, respectively, the inequality relation on the set of integers $\{0, 1, 2\}$, the variables and the constraints of the instance described in Example 1, and finally the instance itself. In general, a definable instance may use parameters. □

B. Group actions, equivariant sets and orbits

Recall that a group G *acts* on a set U if a mapping $G \times U \to U$ is provided, denoted $(\pi, u) \mapsto \pi \cdot u$, such that $1 \cdot u = u$ and $(\pi \cdot \sigma) \cdot u = \pi \cdot (\sigma \cdot u)$, for all $\pi, \sigma \in G, u \in U$, and 1 the identity element in G. An *orbit* under this action is any set of the form $\{\pi \cdot u : \pi \in G\}$, where $u \in U$.

Let $\mathrm{Aut}(\mathcal{A})$ denote the group of *atom permutations*, i.e., bijections $\pi : \mathcal{A} \to \mathcal{A}$. If $\mathrm{Aut}(\mathcal{A})$ acts on a set U, then we

[2] Tuples and integers could be encoded by standard set-theoretic tricks, but to improve readability we refrain from that. To interpret unions of non-sets we may treat the latter as singleton sets.

say that U is *equivariant*. Note that every equivariant set is the disjoint union of its orbits.

The group $\mathrm{Aut}(\mathcal{A})$ naturally acts on the set $\mathcal{D}$ of definable sets. Indeed, if $\pi \in \mathrm{Aut}(\mathcal{A})$ and X is a set defined by an expression e and valuation val, then let

$$\pi \cdot X = \pi \cdot e[\mathrm{val}] \stackrel{\text{def}}{=} e[\mathrm{val}; \pi],$$

where $\mathrm{val}; \pi$ denotes function composition (i.e., $(\mathrm{val}; \pi)(v) = \pi(\mathrm{val}(v))$). This is well defined: it is easy to prove by induction on expressions that $e[\mathrm{val}] = e'[\mathrm{val}']$ implies $e[\mathrm{val}; \pi] = e'[\mathrm{val}'; \pi]$. For example, if $\pi(\underline{1}) = \underline{2}$ and $\pi(\underline{2}) = \underline{3}$, then $\pi \cdot (\underline{1}, \underline{2}) = (\underline{2}, \underline{3})$ and $\pi \cdot \mathcal{A}^2 = \mathcal{A}^2$.

As a result, the group $\mathrm{Aut}(\mathcal{A})$ acts on $\mathcal{D}$. Moreover, $x \in X$ implies $\pi \cdot x \in \pi \cdot X$, for any $\pi \in \mathrm{Aut}(\mathcal{A})$ and definable sets x, X. We say that a definable set X is equivariant if $\pi \cdot X = X$ for all $\pi \in \mathrm{Aut}(\mathcal{A})$. This is consistent with the previous definition of equivariance, since $\mathrm{Aut}(\mathcal{A})$ then acts on X. The *orbits* of X are its orbits under the action of $\mathrm{Aut}(\mathcal{A})$. We say that X is *orbit-finite* if X has finitely many orbits.

The sets $\mathcal{A}$, $\mathcal{A}^n$ and $\mathcal{A}^{(n)}$, and the function s in Example 8 are equivariant. All sets in Example 9 are equivariant.

We will use the following results describing the action of $\mathrm{Aut}(\mathcal{A})$ on definable sets. All these results follow, using standard model-theoretic techniques (see e.g. [25]), from the evident fact the structure $\mathbb{A} = (\mathcal{A}, =)$ is *homogeneous*, i.e., every isomorphism between two finite substructures of $\mathbb{A}$ extends to an automorphism of $\mathbb{A}$. In particular, it admits *quantifier-elimination*, i.e., every first-order formula is equivalent to a quantifier-free formula.

Theorem 10. *Let X be a set definable by an expression without free variables. Then:*

1) *X is equivariant.*
2) *X is definable by a quantifier-free expression which can be computed in polynomial space from any expression defining X.*
3) *X is orbit-finite, and each of its orbits is definable by a quantifier-free expression.*
4) *If Y is an equivariant subset of X, then Y is definable by a quantifier-free expression.*

Proof: see Appendix A1. □

Proposition 11. *For definable sets X and Y, it is decidable in polynomial space whether $X \in Y$, $X \subseteq Y$, $X = Y$.*

Proof: see Appendix A2. □

A *system of orbit representatives* of X is a set R which contains exactly one element of each orbit of X.

Lemma 12. *One can compute, in polynomial space, a system of orbit representatives of a definable set X.*

Proof: see Appendix A3. □

C. Order-definable sets

We will sometimes find it advantageous to have a total order $<$ on $\mathcal{A}$, isomorphic to the ordering of the rational numbers. The development in Section II-A can be extended to this setting, by the use of the total order relation $<$ in first-order formulas that define sets with atoms. Sets and functions defined this way will be called *order-definable*. Clearly, any definable set is also order-definable. When we want to make clear that we do not allow the total order in the formulas, we can speak about *equality-definable* sets.

Example 13. The total order relation $< \subseteq \mathcal{A}^2$ is order-defined by the expression

$$\{(a, b) \mid a, b \in \mathcal{A}, a < b\}.$$

In contrast, no total ordering on $\mathcal{A}$ is equality-definable. □

Let $\mathrm{Aut}(\mathcal{A}, <)$ be the group of *monotone atom permutations*, i.e., automorphisms of the total order on $\mathcal{A}$. The notions of action, equivariance and orbits, can be developed as previously, with $\mathrm{Aut}(\mathcal{A})$ replaced by $\mathrm{Aut}(\mathcal{A}, <)$ throughout, and definable sets replaced by order definable sets. (This is a special case of a construction from [23], where various "atom symmetries" are considered, which is itself a special case of permutation models studied in set theory.) To distinguish from previous notions, we shall speak of *monotone-equivariant* sets and functions.

Note that the restriction to monotone permutations may cause the number of orbits in a set to grow. For example, the set $\mathcal{A}^{(n)}$, a single-orbit set under the action of $\mathrm{Aut}(\mathcal{A})$, decomposes into $n!$ orbits under the action of $\mathrm{Aut}(\mathcal{A}, <)$. However, an orbit-finite set remains orbit-finite under the action of monotone atom permutations.

All results from Section II-B hold for order-definable sets as well. For this, it is crucial that the structure $(\mathbb{Q}, <)$ is homogeneous. Indeed, for a different order such as that of natural or integer numbers, most of those results would fail.

III. Infinite instances

In this section we study the decidability and complexity of solving definable CSP instances with atoms, i.e., the decision problem CSP-Inf($\mathbb{T}$) described in the Introduction. Examples 1 and 2 only concern finite templates; we prove decidability for the more general case of *locally finite* templates, where every relation is finite.

In Section III-A the main result is Theorem 19, which states that the existence of a solution of a definable instance over a locally finite template can be decided. The key technical step towards it, Theorem 17, will also be of use in further sections. In Section III-B we focus on the case when the template $\mathbb{T}$ is finite, and prove matching lower and upper bounds on the complexity of the problem CSP-Inf($\mathbb{T}$).

Before we begin, observe that for an equivariant instance, the existence of a solution does not imply the existence of an equivariant solution.

Example 14. Consider an instance $\mathbb{I}$ with $\mathcal{A}^{(2)}$ as the set of variables, and with the set of constraints:

$$\big\{\big(((a, b), (b, a)), R\big) \mid a, b \in \mathcal{A}, a \neq b\big\},$$

where $R = \{(0,1),(1,0)\}$ is the inequality relation on the finite domain $\{0,1\}$. It is easy to see that $\mathbb{I}$ has a solution. Indeed, for any distinct atoms a and b, one can arbitrarily assign a value 0 or 1 to the variable (a,b), and then assign the other value to (b,a). However, it is impossible to do so in a way that would be invariant with respect to all atom permutations, therefore no equivariant solution exists.

On the other hand, there exists a monotone-equivariant solution, namely the assignment

$$f(a,b) = \begin{cases} 0 & \text{if } a < b \\ 1 & \text{otherwise.} \end{cases}$$

This anticipates Theorem 17. □

A. General decidability

For any instance $\mathbb{I}$ over a template $\mathbb{T}$, let $\hom(\mathbb{I},\mathbb{T})$ denote the set of solutions of $\mathbb{I}$. It is a subset of the set T^I of all functions from I to T. The latter set is equipped with the product topology, i.e., where basic open neighborhoods of a function $f : I \to T$ are of the form:

$$\mathcal{B}_J(f) = \{g : I \to T \mid g(x) = f(x) \text{ for all } x \in J\} \qquad (1)$$

for finite subsets $J \subseteq I$. This topology is also called the topology of *pointwise convergence* (where T is discrete), since a sequence of mappings $f_1, f_2, \ldots$ converges in this topology if and only if the sequence $f_1(v), f_2(v), \ldots$ stabilizes for every $v \in I$.

The following lemma extracts a crucial property of locally finite templates. We say that an instance is *constrained* if every variable in it appears in some constraint.

Lemma 15. *For any locally finite template $\mathbb{T}$, and any constrained instance $\mathbb{I}$ over $\mathbb{T}$, $\hom(\mathbb{I},\mathbb{T})$ is a compact subset of T^I.*

Proof: see Appendix B1. □

Similarly as above, the group $\mathrm{Aut}(\mathcal{A})$ of atom permutations inherits the structure of a topological space (in fact, a topological group, which is not even locally compact), as a subset of $\mathcal{A}^{\mathcal{A}}$ with the product topology.

Lemma 16. *For any definable, equivariant, constrained instance $\mathbb{I}$ over an equivariant template $\mathbb{T}$, $\mathrm{Aut}(\mathcal{A})$ acts continuously on $\hom(\mathbb{I},\mathbb{T})$.*

Proof: see Appendix B2. □

By definition, fixpoints of the action of $\mathrm{Aut}(\mathcal{A})$ on $\hom(\mathbb{I},\mathbb{T})$ are exactly equivariant solutions. Example 14 shows that the action may have no fixpoints. This changes if we assume $\mathcal{A}$ to be ordered and restrict to monotone atom permutations, as described in Section II-C. Indeed:

Theorem 17. *For any definable, equivariant, constrained instance $\mathbb{I}$ over an equivariant, locally finite template $\mathbb{T}$, if $\mathbb{I}$ has a solution than it has a monotone-equivariant solution.*

Proof. Note that $\mathrm{Aut}(\mathcal{A},<)$ is isomorphic to the automorphism group of the total order of rational numbers. It is also a subgroup of $\mathrm{Aut}(\mathcal{A})$, so by Lemma 16 it acts continuously on $\hom(\mathbb{I},\mathbb{T})$. By definition, the fixpoints of this action are exactly monotone-equivariant solutions. Now apply Pestov's theorem (Theorem 4) using Lemma 15. □

The last missing part for the decidability of CSP-Inf($\mathbb{T}$) is the following lemma, whose proof follows general principles of equivariant computation on orbit-finite structures for arbitrary atom symmetries, studied in [22], [26].

Lemma 18. *For any equivariant, locally finite template $\mathbb{T}$, it is decidable whether a given definable, constrained, equivariant instance $\mathbb{I}$ over $\mathbb{T}$ has a monotone-equivariant solution.*

Proof: see Appendix B3. □

Finally, we are ready to prove the main result of this section:

Theorem 19. *For any equivariant, locally finite template $\mathbb{T}$, it is decidable whether a given definable, equivariant $\mathbb{I}$ over $\mathbb{T}$ has a solution.*

Proof. Remove the variables of $\mathbb{I}$ which are not constrained and apply Theorem 17 and Lemma 18. □

Remark 20. The careful reader may notice that Example 2 from the Introduction does not quite fit the development presented so far. Indeed, the instance (i.e. the equation system) considered there is not equivariant, or definable by an expression without free variables: atoms $\underline{1}$ and $\underline{2}$ are singled out in it. However, it is definable by an expression with two free variables, say v_1, v_2, with a valuation val that maps them to $\underline{1}$ and $\underline{2}$ respectively. In terminology of [21], [23], the instance is *supported* by the set $\{\underline{1},\underline{2}\}$.

With a little effort, the results of this section can be generalized to all definable instances and templates, dropping the equivariance assumption. Indeed, let both $\mathbb{I}$ and $\mathbb{T}$ be definable by expressions with free variables and valuations, with other assumptions as before. Let $a_1, \ldots, a_n \in \mathcal{A}$ be the (finite) set of all atoms taken as values by either of the valuations. Lemma 15 holds with no change. Lemma 16 holds as well, with $\mathrm{Aut}(\mathcal{A})$ replaced by $\mathrm{Aut}(\mathcal{A}, a_1, \ldots, a_n)$, the group of those atom permutations that fix all the a_i.

Consider the group $\mathrm{Aut}(\mathcal{A}, <, a_1, \ldots, a_n)$ of those *monotone* atom permutations that fix all the a_i; by analogy to the equivariant case, this group acts continuously on solutions of $\mathbb{I}$, and its fixpoints are monotone solutions invariant with respect to all permutations in that group.

To re-prove Theorem 17, notice that $\mathrm{Aut}(\mathcal{A}, <, a_1, \ldots, a_n)$ is an open subgroup of $\mathrm{Aut}(\mathcal{A}, <)$ therefore, by [27, Lemma 13], Theorem 4 works for it as well. Finally, Lemma 18 and Theorem 19 are proved entirely analogously for all definable structures.

A more substantial generalization is also possible, where one replaces a mere set $\mathcal{A}$ by a *homogeneous* relational structure of atoms, along the lines of [23]. Supposing that $\mathcal{A}$ is a reduct of a so-called Ramsey structure with enough decidability properties, the above development can be repeated, with Pestov's theorem replaced by its generalization due to Kechris, Pestov and Todorcevic [7]. A more detailed description of this is deferred to a full version of this paper.

B. Finite templates

Consider now a classical, *finite* template $\mathbb{T}$, without atoms, such as in Examples 1 and 2. The algorithm for solving definable instances over $\mathbb{T}$ that arises from a general proof of Lemma 18 is double exponential. However, for finite templates this complexity can be lowered using the following PSPACE reduction to CSP($\mathbb{T}$):

Proposition 21. *For every equivariant, definable instance $\mathbb{I}$ over a finite template $\mathbb{T}$ one can compute (in polynomial space) a finite instance $\mathbb{I}^*$ of size exponential in the size of the set expression that defines $\mathbb{I}$, and such that $\mathbb{I}$ has a solution if and only if $\mathbb{I}^*$ has a solution.*

Proof. By Theorem 17, $\mathbb{I}$ has a solution if and only if it has a monotone-equivariant solution. We construct a finite instance $\mathbb{I}^*$ whose solutions correspond bijectively to monotone-equivariant solutions of $\mathbb{I}$. In the rest of the proof, by orbits we mean orbits with respect to monotone atom permutations.

Let e be the set expression that defines the instance $\mathbb{I}$. The set I^* of variables of $\mathbb{I}^*$ consists of the orbits of the set I of variables of $\mathbb{I}$. Their number is at most exponential in the size of e and they can be enumerated in polynomial space, by scanning all quantifier-free types of formulas with n free variables, where n is the number of variables in the expression e.

For every constraint $\big((x_1, \dots, x_k), R\big)$ of $\mathbb{I}$ we take the orbits $O_1, \dots, O_k$ of the variables $x_1, \dots, x_k$, respectively, and add a constraint $\big((O_1, \dots, O_k), R\big)$ to $\mathbb{I}^*$. The number of constraints is also at most exponential in the size of e.

Every monotone-equivariant function from I to T is constant on the orbits of I. Hence, it is easy to see that there is a bijective correspondence between solutions of $\mathbb{I}^*$ and monotone-equivariant solutions of $\mathbb{I}$. □

By analogy to Remark 20, the above result can be generalized to all definable instances, and hence gives an upper complexity bound of solving definable instances for each finite template $\mathbb{T}$: if CSP($\mathbb{T}$) is in a complexity class $\mathcal{C}$ such that $\mathrm{L} \subseteq \mathcal{C}$, then CSP-Inf($\mathbb{T}$) is in the "exponentially larger" class $\exp(\mathcal{C})$, defined by:

$$\exp(\mathcal{C}) = \bigcup_{k \in \mathbb{N}} \{L : \mathrm{pad}_k(L) \in \mathcal{C}\}$$

where, for a language L, the language $\mathrm{pad}_k(L)$ consists of words from L of any length m padded with arbitrary letters to the length 2^{m^k}. For example, $\exp(\mathrm{L}) = \mathrm{PSPACE}$ and $\exp(\mathrm{NP}) = \mathrm{NEXP}$.

As it turns out, this upper bound is tight:

Theorem 22. *For any $\mathcal{C}$ such that $\mathrm{L} \subseteq \mathcal{C}$, if* CSP($\mathbb{T}$) *is $\mathcal{C}$-hard then* CSP-Inf($\mathbb{T}$) *is $\exp(\mathcal{C})$-hard, under logarithmic space reductions.*

Proof: see Appendix B4. □

Together with Proposition 21, this proves Theorem 3. As examples, 3-colorability of finite graphs is NP-complete, so the same problem for definable graphs (see Example 1) is NEXP-complete; solving finite systems of linear equations over $\mathbb{Z}_k$ is $\mathrm{MOD}_k\mathrm{L}$-complete [28], so the same problem for definable systems of equations (see Example 2) is complete for the class $\exp(\mathrm{MOD}_k\mathrm{L}) = \mathrm{MOD}_k\mathrm{PSPACE}$ of those languages L for which there exists a nondeterministic polynomial space Turing machine M_L such that $w \in L$ if and only if the number of accepting runs of M_L on input w is divisible by k.

Remark 23. Our proof of Theorem 22 actually works already if CSP($\mathbb{T}$) is $\mathcal{C}$-complete under poly-logarithmic space reductions; completeness of $\mathcal{C}$ under logarithmic space reductions is not necessary. Note that it is an open problem whether the class of poly-log space algorithms is contained in PTIME.

In a proof of Theorem 22 one must construct definable instances of CSP-Inf($\mathbb{T}$) from words in a language $L \in \exp(\mathcal{C})$. In our general proof, set expressions that define these instances use the full power of first order logic, including quantification over atoms. However, for all standard examples of templates $\mathbb{T}$, e.g. those that correspond to the 3-colorability problem, Boolean 3-satisfiability, solving linear equations over $\mathbb{Z}_k$ or Boolean Horn-satisfiability, we have found reductions that use the simplest possible formulas over atoms: comparing two atoms for equality. We are unable to find a general proof which would only use so simple set expressions.

IV. FINITE INSTANCES

In this section we study the complexity of the problem CSP($\mathbb{T}$), where $\mathbb{T}$ is a fixed locally finite template. Recall that instances of this problem are finite structures. We demonstrate that many results known from the classical setting, where the templates are finite, lift to the locally finite setting. In Section IV-A we show that polymorphisms of $\mathbb{T}$ determine the complexity of CSP($\mathbb{T}$). In Section IV-B we present a general method of lifting results concerning finite templates to locally finite templates, and we show a few applications. Finally, in Section IV-C, we show that for a locally finite, definable template $\mathbb{T}$, the problem CSP($\mathbb{T}$) reduces (in polynomial time) to the problem CSP($\hat{\mathbb{T}}$) for a *finite* template $\hat{\mathbb{T}}$. Moreover, if $\mathbb{T}$ admits polymorphisms satisfying some linear identities, then so does $\hat{\mathbb{T}}$, which in many cases gives an upper bound on the complexity of CSP($\mathbb{T}$). To prove these results, we are sometimes forced to consider infinite templates $\mathbb{T}$ as CSP instances over other templates, and there the results of Section III come handy.

A. Algebraic foundations

We now generalize, to locally finite templates, several classical theorems concerning the relationship between the set of polymorphisms of a template, its set of pp-definable relations and its complexity, and constructions of core templates. Most of the proofs are standard, but they require a few tweaks and nontrivial observations.

1) Pp-formulas: A *primitive positive formula* (or *pp-formula*) is a first-order formula, possibly with free variables, which uses only existential quantification (no universal quantifiers), conjunctions (no disjunctions nor negations), and atomic

formulas (no negations of atomic formulas), which might include equalities among variables. A typical pp-formula is

$$\phi(x,z,t) = \exists y.R(x,y) \wedge S(y,z) \wedge z = t.$$

If $\mathbb{T}$ is a relational structure and ϕ is a pp-formula over the signature of $\mathbb{T}$ with free variables $x_1,\dots,x_n$, then we denote by $\phi(\mathbb{T}) \subseteq T^n$ the set of those tuples $(t_1,\dots,t_n)$ which satisfy ϕ in $\mathbb{T}$. Any relation of the form $\phi(\mathbb{T})$, where ϕ is some pp-formula, is said to be a *pp-definable* relation of $\mathbb{T}$. By $\mathrm{ppDef}\,\mathbb{T}$ we denote the family of all *finite* pp-definable relations of $\mathbb{T}$.

A *generalized pp-formula* Φ is a pair $(\mathbb{I},\alpha)$, where $\mathbb{I}$ is an instance (equivalently, a relational structure) and $\alpha : \{x_1,\dots,x_n\} \to I$ is a function from a finite set of *free variables* of Φ. We say that Φ is *finite* if the instance $\mathbb{I}$ has finitely many variables and finitely many constraints. If $\mathbb{T}$ is a relational structure over the same signature as $\mathbb{I}$, then Φ defines a relation on T of arity n:

$$\phi(\mathbb{T}) \stackrel{\text{def}}{=} \big\{\big(f(x_1),\dots,f(x_n)\big) \mid f \in \hom(\mathbb{I},\mathbb{T})\big\}.$$

It is a standard result that finite generalized pp-formulas define the same relations as pp-formulas. The following lemma shows that for finite relations and locally finite templates, *arbitrary* generalized pp-formulas do not define anything more:

Lemma 24. *Let $\mathbb{T}$ be a locally finite template. The following conditions are equivalent for a finite relation $R \subseteq \mathbb{T}^n$*

1) *There is a pp-formula ϕ such that $\phi(\mathbb{T}) = R$,*
2) *There is a finite generalized pp-formula Φ such that $\Phi(\mathbb{T}) = R$.*
3) *There is a generalized pp-formula Φ such that $\Phi(\mathbb{T}) = R$.*

Proof: see Appendix C1. □

Lemma 24 relies on the following lemma, which is a variation of the compactness argument used in Lemma 15 (the union of structures is taken vertex-wise and edge-wise).

Lemma 25. *For any structure $\mathbb{I}$ and an ascending sequence $\mathbb{I}_0 \subseteq \mathbb{I}_1 \subseteq \mathbb{I}_2 \subseteq \cdots \subseteq \mathbb{I}$ of substructures such that $\bigcup_i \mathbb{I}_i = \mathbb{I}$, and for any locally finite structure $\mathbb{T}$, there is a homomorphism $f : \mathbb{I} \to \mathbb{T}$ if and only if for each $n \geq 0$ there is a homomorphism $f_n : \mathbb{I}_n \to \mathbb{T}$. If all the f_n extend a single homomorphism $f_0 : \mathbb{I}_0 \to \mathbb{T}$, then so does f.*

Proof: see Appendix C2. □

Proposition 26. *Let $\mathbb{B}$ and $\mathbb{C}$ be locally finite, definable templates over the same domain B. If* $\mathrm{ppDef}\,\mathbb{B} \subseteq \mathrm{ppDef}\,\mathbb{C}$ *then* CSP($\mathbb{B}$) *reduces to* CSP($\mathbb{C}$)*, via a polynomial-time reduction.*

Proof: see Appendix C3. □

2) The Inv-Pol connection: An *operation* on T is a function $f : T^k \to T$, for some number k (the *arity*). We say that the operation f *preserves* a relation R of arity n on T if for any k tuples in R:

$$(x_{11},\dots,x_{1n}),(x_{21},\dots,x_{2n}),\dots,(x_{k1},\dots,x_{kn}) \in R,$$

the tuple obtained from them by applying f componentwise:

$$\big(f(x_{11},\dots,x_{k1}),f(x_{12},\dots,x_{k2}),\dots,f(x_{1n},\dots,x_{kn})\big)$$

belongs to R as well. We also say that R is *invariant* under the operation f.

If $\mathcal{F}$ is a family of operations of T, then by $\mathrm{Inv}\,\mathcal{F}$ we denote the set of relations which are invariant under every operation in $\mathcal{F}$. Dually, if $\mathcal{R}$ is a family of relations on T, then by $\mathrm{Pol}\,\mathcal{R}$ we denote the set of those operations that preserve all relations in $\mathcal{R}$. If $\mathbb{T}$ is a relational structure, then $\mathrm{Pol}\,\mathbb{T}$ is defined as $\mathrm{Pol}\,\mathcal{R}$, where $\mathcal{R}$ is the set of relations of $\mathbb{T}$, and elements of $\mathrm{Pol}\,\mathbb{T}$ are called *polymorphisms* of $\mathbb{T}$.

A polymorphism of $\mathbb{T}$ of arity n can be equivalently defined as follows. Equip $\mathbb{T}^n$ with the *cartesian product* structure, i.e., for a relation R of $\mathbb{T}$ of arity k and a k-tuple of n-tuples of $\mathbb{T}$,

$$R\big((x_{11},\dots,x_{1n}),(x_{21},\dots,x_{2n}),\dots,(x_{k1},\dots,x_{kn})\big)$$

holds in $\mathbb{T}^n$ if and only if each $i = 1,\dots,n$, the relation $R(x_{1i},x_{2i},\dots,x_{ki})$ holds in $\mathbb{T}$. It is easy to see that a mapping $f : T^n \to T$ is a polymorphism if and only if it is a homomorphism from $\mathbb{T}^n$ to $\mathbb{T}$.

Proposition 27. *For any locally finite template $\mathbb{T}$, and any finite relation R over T, $R \in \mathrm{ppDef}\,\mathbb{T}$ if and only of $R \in \mathrm{Inv}\,\mathrm{Pol}\,\mathbb{T}$.*

Proof: see Appendix C4. □

Proposition 27 is analogous to a fundamental theorem of algebraic finite CSP theory [1]. In the proof of the "only if" part, we define an invariant relation using a generalized pp-formula, and apply Lemma 24.

Theorem 28. *For any two locally finite, definable templates $\mathbb{B}$ and $\mathbb{C}$ over the same domain B, if $\mathrm{Pol}\,\mathbb{C} \subseteq \mathrm{Pol}\,\mathbb{B}$ then* CSP($\mathbb{B}$) *reduces to* CSP($\mathbb{C}$) *via a polynomial-time reduction.*

Proof. If $\mathrm{Pol}\,\mathbb{C} \subseteq \mathrm{Pol}\,\mathbb{B}$ then $\mathrm{Inv}\,\mathrm{Pol}(\mathbb{C}) \supseteq \mathrm{Inv}\,\mathrm{Pol}(\mathbb{B})$, hence $\mathrm{ppDef}(\mathbb{C}) \supseteq \mathrm{ppDef}(\mathbb{B})$. Conclude using Proposition 26. □

Corollary 29. *If $\mathrm{Pol}\,\mathbb{B} = \mathrm{Pol}\,\mathbb{C}$ then* CSP($\mathbb{B}$) *and* CSP($\mathbb{C}$) *are equivalent, up to polynomial time reductions.*

3) Core structures: We say that a structure $\mathbb{T}$ is a *core* if every endomorphism of $\mathbb{T}$ is a monomorphism. If $\mathbb{T}$ is a structure and A is a finite subset of its domain, then by $\mathbb{T}|_A$ we denote the template with domain A, whose relations are the restrictions to A of all relations $R \in \mathrm{ppDef}\,\mathbb{T}$. In the proof of the implication (2→1), again we invoke Pestov's theorem (Theorem 4).

Proposition 30. *Let $\mathbb{T}$ be a monotone-equivariant, locally finite structure without isolated nodes. Then the following conditions are equivalent:*

1) *$\mathbb{T}$ is a core.*
2) *Every monotone-equivariant endomorphism of $\mathbb{T}$ is a monomorphism.*
3) *For any finite set $A \subseteq T$, the structure $\mathbb{T}|_A$ is a finite core.*

Proof: see Appendix C5. □

By analogy to the development in Section III, thanks to Proposition 30 template cores are computable:

Proposition 31. *Given a definable, equivariant, locally finite template $\mathbb{T}$, one can effectively test whether $\mathbb{T}$ is a core, and effectively construct a core which is a retract of $\mathbb{T}$.*

Proof (sketch). A *retraction* of $\mathbb{T}$ is an endomorphism r whose twofold composition $r; r$ is equal to r.

Test all monotone-equivariant retractions of $\mathbb{T}$. If there is such a retraction r which is not onto, then $\mathbb{T}$ is not a core. In this case, replace $\mathbb{T}$ by the image of r and repeat. □

B. From finite to locally finite templates

In this section, we present a general method of lifting results concerning finite templates to locally finite ones. This is achieved by observing that $\mathbb{T}$ can be covered by a family $\mathcal{F}$ of finite subtemplates (defined below), such that the question whether $\mathbb{T}$ admits polymorphisms satisfying a set of linear equations boils down to the question whether each finite template in $\mathcal{F}$ admits such polymorphisms.

Important classes of polymorphisms are defined using sets of identities that they satisfy; usually, those identities are of a specific shape. Formally, let Γ be a functional signature, i.e., a set of function names with associated finite arities. A *linear identity* is an expression of the form $r \approx s$, where r and s are either variables or Γ-terms with exactly one function symbol. A Γ-algebra *satisfies* an identity $r \approx s$ if for any valuation of the variables in r and s, both sides have the same value. For a set E of linear identities, we say that a template $\mathbb{T}$ *admits E-polymorphisms* if there is a Γ-algebra with universe T, whose operations are polymorphisms of $\mathbb{T}$ and which satisfies all identities in E.

For example, a *majority polymorphism* is a ternary polymorphism t that satisfies linear identities:

$$t(x, y, y) \approx t(y, x, y) \approx t(x, y, y) \approx y,$$

and a *Maltsev polymorphism* is a ternary one that satisfies linear identities:

$$t(x, y, y) \approx t(y, y, x) \approx x.$$

Other polymorphism classes defined by linear identities will be considered below.

Let $\mathbb{T} = (T, \mathcal{R})$ be a template and let $\mathcal{R}_0 \subseteq \mathcal{R}$ be some family of relations. The *subtemplate* of $\mathbb{T}$ induced by $\mathcal{R}_0$ is the template $\mathbb{T}_0 = (T_0, \mathcal{R}_0)$ whose domain T_0 consists of all those $x \in T$ which appear in some tuple that belongs to any of the relations in $\mathcal{R}_0$. We define a union $\mathbb{T}_1 \cup \mathbb{T}_2$ of two subtemplates $\mathbb{T}_1 = (T_1, \mathcal{R}_1)$ and $\mathbb{T}_2 = (T_2, \mathcal{R}_2)$ of $\mathbb{T}$ to be the template $(T_1 \cup T_2, \mathcal{R}_1 \cup \mathcal{R}_2)$, and analogously for unions of larger families of subtemplates.

Theorem 32. *Let $\mathbb{T}$ be a locally finite template and let $\mathbb{T}_1 \subseteq \mathbb{T}_2 \subseteq \ldots$ be an ascending sequence of subtemplates of $\mathbb{T}$ such that $\bigcup_i \mathbb{T}_i = \mathbb{T}$. If E is a set of linear identities then $\mathbb{T}$ admits E-polymorphisms if and only if for each i, the template $\mathbb{T}_i$ admits E-polymorphisms.*

Proof. The left-to-right implication is obvious, since any polymorphism of $\mathbb{T}$ is a polymorphism of all its subtemplates. To show the other implication we use a standard construction which allows to express a polymorphism of a template $\mathbb{T}$ as a solution of a CSP instance. Let E be a set of identities using function symbols from a functional signature Γ. We assume that there are no identities of the form $x \approx y$ where both x, y are either variables or constants, as the general case can be easily reduced to this one. For a template $\mathbb{T}$, define an instance $\mathbb{F}_E(\mathbb{T})$ as a disjoint union of instances as follows:

$$F_E(\mathbb{T}) = \coprod_{g \in \Gamma} \mathbb{T}_g,$$

where $\mathbb{T}_g$ is the n-fold cartesian product of $\mathbb{T}$, for n the arity of g. Furthermore, extend $F_E(\mathbb{T})$ by constraints as described below. For each identity in E, consider two possibilities:

- If the identity is of the form $g(\bar{x}) \approx g'(\bar{y})$, with $g, g' \in \Gamma$, then for every valuation $\mathrm{val} : V \to \mathbb{T}$ of the variables V in the tuples $\bar{x}$ and $\bar{y}$, add a binary constraint $\big((\bar{x}[\mathrm{val}], \bar{y}[\mathrm{val}]), =\big)$, where $\bar{x}[\mathrm{val}] \in \mathbb{T}_g$ and $\bar{y}[\mathrm{val}] \in \mathbb{T}_{g'}$.
- If the identity is of the form $g(\bar{x}) \approx u$ or $u \approx g(\bar{x})$, with $g \in \Gamma$ and u a variable or a constant, then for each valuation $\mathrm{val} : V \to \mathbb{T}$ of the variables V in $\bar{x}$ and u, add a unary constraint $\big(\bar{x}[\mathrm{val}], \{u[\mathrm{val}]\}\big)$.

Let $\overline{\mathbb{T}}$ be the structure $\mathbb{T}$ equipped additionally with the singleton unary relations and the relation $=$. The instance $F_E(\mathbb{T})$ is over $\overline{\mathbb{T}}$, and it has a solution if and only if $\mathbb{T}$ admits E-polymorphisms.

Suppose now that $\mathbb{T}$ is locally finite and let $\mathbb{T}_1 \subseteq \mathbb{T}_2 \subseteq \ldots$ be an ascending sequence of subtemplates of $\mathbb{T}$ such that $\bigcup_i \mathbb{T}_i = \mathbb{T}$, and for each i the template $\mathbb{T}_i$ admits E-polymorphisms. This means that for every i there exists a solution of the instance $F_E(\mathbb{T}_i)$ (which can be seen as an instance over $\overline{\mathbb{T}}$). The corresponding sequence of instances $(F_E(\mathbb{T}_i))_i$ is ascending and satisfies $\bigcup F_E(\mathbb{T}_i) = F_E(\mathbb{T})$. It follows from Lemma 25 that there exists a solution of the instance $F_E(\mathbb{T})$. Hence $\mathbb{T}$ admits E-polymorphisms. □

Remark 33. Note that if E is a finite set of linear identities and the template $\mathbb{T}$ is definable, then so is $F_E(\mathbb{T})$. By Theorem 19, it can be decided whether $\mathbb{T}$ admits E-polymorphisms.

In the literature, there are two general algorithmic principles for solving CSP($\mathbb{T}$) in polynomial time for wide classes of finite templates $\mathbb{T}$. Theorem 32 lets us translate algebraic characterizations of those classes to the setting of locally finite templates.

The first algorithm can be seen as a generalization of Gaussian elimination, and it is based on the fact that for every finite instance $\mathbb{I}$ over $\mathbb{T}$ the set of solutions has a "small" representing set. For a finite structure $\mathbb{A}$ let $s_{\mathbb{A}}(n)$ denote the logarithm, base 2, of the number of all different pp-definable relations of arity n in $\mathbb{A}$. We say that a template $\mathbb{T}$ has *few subpowers* if there is a natural number k such that for every

finite subtemplate $\mathbb{B}$ of $\mathbb{T}$ we have that $s_{\mathbb{B}}(n)$ is bounded from above by a polynomial of degree k. It is known [29], [30] that a finite template $\mathbb{T}$ has few subpowers if and only if $\mathbb{T}$ admits a k-edge polymorphism, where a *k-edge* polymorphism e is a $k+1$-ary polymorphism that satisfies the identities

$$e(x,x,y,y,y,\dots,y,y) \approx e(x,y,x,y,y,\dots,y,y) \approx y$$

and

$$e(y,y,y,x,y,\dots,y,y) \approx e(y,y,y,y,x,\dots,y,y) \approx \dots$$
$$\dots \approx e(y,y,y,y,y,\dots,x,y) \approx e(y,y,y,y,y,\dots,y,x) \approx y.$$

Corollary 34. *A locally finite template $\mathbb{T}$ has few subpowers if and only if $\mathbb{T}$ admits a k-edge polymorphism for some $k > 0$.*

Moreover, k-edge polymorphisms characterize in some sense (see [29]) all locally finite templates that can by solved in polynomial time by a "Gaussian-like" algorithm.

The second algorithm determines whether a solution of a given instance exists by looking at subinstances of bounded size and checking if there is a consistent set of local solutions. A template of *bounded-width* is a template for which such an algorithm, called the *local consistency algorithm*, is correct. It follows from the results of [31]–[33] that a core finite template $\mathbb{T}$ has bounded width if and only if $\mathbb{T}$ admits weak near unanimity polymorphisms v of arity 3 and w of arity 4 that satisfy $v(y,x,x) \approx w(y,x,x,x)$, where a *weak near unanimity* polymorphism t is one that satisfies the identities

$$t(x,x,\dots,x) \approx x \text{ and}$$
$$t(y,x,\dots,x) \approx t(x,y,x,\dots,x) \approx \dots \approx t(x,x,\dots,x,y).$$

Corollary 35. *A core, locally finite template $\mathbb{T}$ has bounded width if and only if $\mathbb{T}$ admits weak near unanimity polymorphisms v of arity 3 and w of arity 4 that satisfy $v(y,x,x) \approx w(y,x,x,x)$. Moreover, it can be decided whether a definable, locally finite template $\mathbb{T}$ has bounded width.*

Proof (sketch). We show the harder "only if" part. Suppose that $\mathbb{T}$ has bounded width. Consider a finite set $A \subseteq T$. By Proposition 30, $\mathbb{T}|_A$ is a finite core and has bounded width. By the results of finite CSP theory, $\mathbb{T}|_A$ admits polymorphisms v_A, w_A satisfying the required linear identities. Since A is arbitrary, we may apply Theorem 32, obtaining polymorphisms v, w of $\mathbb{T}$, which also satisfy the required identities.

The last part of the statement of the corollary follows from Proposition 31 and Remark 33. □

Purely algebraic results, which show for instance that the existence of polymorphisms of some kind ensures a finite template $\mathbb{T}$ to admit some other polymorphisms can also be translated to the locally finite setting using Proposition 32. The following is an easy consequence of Lemma 9 of [34]:

Corollary 36. *If a core, locally finite template $\mathbb{T}$ has a Maltsev polymorphism and has bounded width, then it has a majority polymorphism.*

Many other results can be proved following the same pattern.

C. From locally finite to finite templates

This section is a proof of the following theorem:

Theorem 37. *For every equivariant, definable, locally finite template $\mathbb{T}$ there is a finite template $\hat{\mathbb{T}}$ such that:*

- *for every finite instance $\mathbb{I}$ over $\mathbb{T}$ there is a finite instance $\hat{\mathbb{I}}$ over $\hat{\mathbb{T}}$:*
  - *computable from $\mathbb{I}$ in polynomial time, and*
  - *such that $\mathbb{I}$ has a solution if and only if $\hat{\mathbb{I}}$ has a solution,*
- *for every set of linear identities E, if $\mathbb{T}$ admits E-polymorphisms then $\hat{\mathbb{T}}$ admits E-polymorphisms.*

We assume atoms to be totally ordered as described in Section II-C. By $\underline{i}$ we denote the atom that corresponds to the rational number i. All orbits below are considered with respect to monotone atom permutations.

Fix any order-definable set $x = e[\mathrm{val}]$, where e is an expression and $\mathrm{val} : V \to \mathcal{A}$ a valuation. There is a unique order-preserving bijection f between the range of val and the set $\{\underline{1}, \dots, \underline{n}\}$, where n is the number of elements in the range of val. By $[x]$ we denote the set defined by $e[\mathrm{val}; f]$. Note that $[x] = [y]$ if and only if they are in the same orbit. The set $[x]$ can therefore be seen as a representative of this orbit.

Assume without loss of generality that all instances are constrained. For any instance $\mathbb{I}$ over $\mathbb{T}$ and a variable $x \in I$, let $\mathbb{I}_x$ be the largest constrained subinstance of $\mathbb{I}$ such that x belongs to the tuple of variables in every constraint of $\mathbb{I}_x$. By $U_{(\mathbb{I},x)}$ we denote the unary predicate over $\mathbb{T}$ defined by the generalized pp-formula $(\mathbb{I}_x, \alpha_x)$, where $\alpha_x : \{x\} \to \mathbb{I}_x$ is the inclusion mapping (we write simply U_x whenever the instance $\mathbb{I}$ is clear from the context). Observe that if $f : \mathbb{I} \to \mathbb{T}$ is a solution of the instance $\mathbb{I}$ then $f(x) \in U_{(\mathbb{I},x)}$. We say that an assignment $f : I \to T$ is *feasible* if it maps every x to an element of $U_{(\mathbb{I},x)}$. To decide whether $\mathbb{I}$ has a solution, it is enough to consider feasible assignments.

Fix a constrained instance $\mathbb{I}$ over $\mathbb{T}$. For each variable $x \in I$, let g_x be a monotone atom permutation that maps U_x to $[U_x]$. We define an instance $\hat{\mathbb{I}}$ as follows:

- the variables of $\hat{\mathbb{I}}$ are the variables of $\mathbb{I}$,
- for every constraint $\big((x_1,\dots,x_n), R\big)$ in $\mathbb{I}$ there is a constraint $\big((x_1,\dots,x_n), \hat{R}\big)$ in $\hat{\mathbb{I}}$, where

$$\hat{R} = g\big(R \cap (U_{x_1} \times \dots \times U_{x_n})\big),$$
$$g(t_1,\dots t_n) = (g_{x_1}(t_1), \dots, g_{x_n}(t_n)).$$

  Note that $\hat{R} \subseteq [U_{x_1}] \times \dots \times [U_{x_n}]$.

It is immediate from the above construction that for any instance $\mathbb{I}$, there is a bijective correspondence between feasible assignments for $\mathbb{I}$ and assignments for $\hat{\mathbb{I}}$: if a feasible assignment f for $\mathbb{I}$ maps a variable x to some $t \in U_x$, then the corresponding assignment for $\hat{\mathbb{I}}$ maps x to $g_x(t) \in [U_x]$. Moreover, a feasible assignment for $\mathbb{I}$ is a solution if and only if the corresponding assignment for $\hat{\mathbb{I}}$ is a solution. It follows that $\mathbb{I}$ has a solution if and only if $\hat{\mathbb{I}}$ has a solution.

We now define a finite template $\hat{\mathbb{T}}$ such that for any instance $\mathbb{I}$ over $\mathbb{T}$, the instance $\hat{\mathbb{I}}$ is over $\hat{\mathbb{T}}$. The domain of $\hat{\mathbb{T}}$ is the union:

$$\hat{T} = \bigcup_{(\mathbb{I},x)} [U_{(\mathbb{I},x)}],$$

where $(\mathbb{I}, x)$ ranges over all constrained instances $\mathbb{I}$ over $\mathbb{T}$, and their variables x. The relations of $\hat{\mathbb{T}}$ are all the relations which appear in the constraints of all instances of the form $\hat{\mathbb{I}}$. Notice that if the $\mathbb{T}$ is definable then the template $\hat{\mathbb{T}}$ is finite. Indeed, since $\mathbb{T}$ has finitely many orbits, the number of elements in the unary predicates $U_{(\mathbb{I},x)}$ is bounded, and hence they form an orbit-finite set. The domain $\hat{T}$ is the sum of their representatives, so it is finite. Finally, the arity of relations in $\hat{\mathbb{T}}$ is bounded by the maximal arity of a relation in $\mathbb{T}$, so there are finitely many possible relations.
This proves half of Theorem 37; we now prove the second half.

Lemma 38. *Let E be a set of linear identities. An equivariant, definable, locally finite template $\mathbb{T}$ admits E-polymorphisms if and only if it admits monotone-equivariant E-polymorphisms.*

Proof. In the proof of Theorem 32, an instance $\mathbb{F}_E(\mathbb{T})$ is constructed whose solutions correspond to E-polymorphisms of $\mathbb{T}$. Let $\mathbb{I}$ be its largest constrained subinstance. By Theorem 17, if $\mathbb{I}$ has a solution then it has a monotone-equivariant solution f. Since the domain of $\mathbb{F}_E(\mathbb{T})$ is a disjoint union of sets of the form T^n, it is possible to extend this solution to a monotone-equivariant solution of $\mathbb{F}_E(\mathbb{T})$, for example by defining $f(x)$ to be the projection to the first coordinate whenever $x \notin \mathbb{I}$. This solution corresponds to a monotone-equivariant E-polymorphism of $\mathbb{T}$. □

We now finish the proof of Theorem 37. For any set of linear identities E, let A be an algebra with universe T, whose operations are polymorphisms of $\mathbb{T}$, and such that A satisfies the identities in E. By Lemma 38 we can assume that the operations of A are monotone-equivariant. We define an algebra $\hat{A}$ with universe $\hat{T}$, whose operations are polymorphisms of $\hat{\mathbb{T}}$, and such that $\hat{A}$ satisfies the identities in E. Observe that the universe $\hat{T}$ is a subset of the universe T. Let $r : T \to \hat{T}$ be a function which is the identity on $\hat{T}$, and maps all the other elements of T to some fixed element t of $\hat{T}$. For every operation f of A, define the corresponding operation $\hat{f}$ of $\hat{A}$ by $\hat{f} = f; r$. Since all identities in E are linear, it is easy to see that $\hat{A}$ satisfies them.

It remains to show that every $\hat{f}$ is a polymorphism of $\hat{\mathbb{T}}$. To this end, pick an n-ary relation $\hat{R}$ of $\hat{\mathbb{T}}$. It follows from the construction of the template $\hat{\mathbb{T}}$ that there exists a pp-definable relation R in $\mathbb{T}$ such that $\hat{R} = F(R)$, where $F : \mathcal{A}^n \to \mathcal{A}^n$ is a tuple of monotone atom permutations. The relation R is invariant under the monotone-equivariant f, hence it is easy to see that so is $\hat{R}$. Moreover, since R is a subset of $\hat{T}^n$, it follows that it is invariant under $\hat{f}$. □

V. Order-invariant logics

In a previous paper [10], we studied the expressive power of various logics over a certain class of structures, which we now recall in a slightly simplified version. As an application of the methods developed in this paper, we briefly say how the main result of [10] can be re-developed in the setting of locally finite CSPs, showing the key ideas of the proof much more clearly and clearing them from a clutter of technicalities.

For a fixed, finite graph $\mathfrak{p}$, a *linearly $\mathfrak{p}$-patched structure* is a finite graph $\mathbb{G}$, together with a linearly ordered family $\mathfrak{p}_1 < \ldots < \mathfrak{p}_n$ of subgraphs of $\mathbb{G}$ (called *patches*), each of which is isomorphic to $\mathfrak{p}$, and which cover $\mathbb{G}$, i.e.,

$$\mathbb{G} = \mathfrak{p}_1 \cup \ldots \cup \mathfrak{p}_n.$$

For example, if $\mathfrak{p}$ is the 2-clique, then a linearly $\mathfrak{p}$-patched structure is the same as a graph together with a linear ordering of its edges.

First order formulas can be evaluated on linearly $\mathfrak{p}$-patched structures, allowing quantification $\forall v, \exists v$ over the vertices, $\forall \mathfrak{q}, \exists \mathfrak{q}$ over the patches, comparison $\mathfrak{q} < \mathfrak{q}'$ of the patches with respect to their linear ordering, and tests $\mathfrak{p} \models E(v, w)$ for each patch $\mathfrak{p}$ and vertices v, w (where E denotes the edge predicate). Various extensions of first order logic can be also evaluated over linearly $\mathfrak{p}$-patched structures, in particular the *Least Fix Point* logic (*LFP*), a well studied extension of first order logic by a fixpoint operation [8], [35]. It turns out that over linearly $\mathfrak{p}$-patched structures, LFP is equivalent to LFP+C (a further extension by a counting mechanism) and to polynomial time *Turing Machines with Atoms* – an analogue of Turing machines in the realms of sets with atoms [10], [24].

On the other hand, classical Turing machines can be evaluated on linearly $\mathfrak{p}$-patched structures, using standard bit-string encodings of relational structures. We say that LFP is equally expressive as PTIME over a class of structures $\mathcal{C}$, if every property of structures in $\mathcal{C}$ which is decidable in polynomial time, can be expressed by a formula of LFP.

The famous Immerman-Vardi [36], [37] and Cai-Fürer-Immerman [8] theorems can be reformulated as follows.

Theorem 39. *If $\mathfrak{p}$ is the 2-clique, then LFP is equally expressive as* PTIME *over linearly $\mathfrak{p}$-patched structures.*

Theorem 40. *There is a graph $\mathfrak{p}$, namely the disjoint union of two 3-cliques, such that LFP+C is less expressive than* PTIME *over linearly $\mathfrak{p}$-patched structures.*

A corollary of the main result of [10] is the following.

Theorem 41. *Given a graph $\mathfrak{p}$, it can be effectively decided whether LFP is equally expressive as* PTIME *over linearly $\mathfrak{p}$-patched structures.*

The proof of the theorem proceeds in several steps, and starts with the following observation.

Lemma 42. *LFP is equally expressive as* PTIME *over linearly $\mathfrak{p}$-patched structures if and only if it can test their isomorphism.*

Consider a pair of linearly $\mathfrak{p}$-patched structures $\mathbb{G}, \mathbb{G}'$, whose vertices are atoms, both with n patches denoted $\mathfrak{p}_1 < \ldots < \mathfrak{p}_n$ and $\mathfrak{q}_1 < \ldots < \mathfrak{q}_n$, respectively. An

isomorphism from $\mathbb{G}$ to $\mathbb{G}'$ must map each $\mathfrak{p}_i$ isomorphically to $\mathfrak{q}_i$, for $i = 1, \ldots, n$. Define a finite instance $\mathbb{I}$ with variables $1, 2, \ldots, n$, and for each $1 \leq i, j \leq n$, a binary constraint

$$((i, j), C^{\mathfrak{p},\mathfrak{p}'}_{\mathfrak{r},\mathfrak{r}'}),$$

such that $C^{\mathfrak{p},\mathfrak{p}'}_{\mathfrak{r},\mathfrak{r}'}$ is the set of pairs (α, β) where $\alpha : \mathfrak{p}_i \to \mathfrak{q}_i, \beta : \mathfrak{p}_j \to \mathfrak{q}_j$ are isomorphisms which are consistent, i.e., $\alpha(v) = \beta(v)$ for all v such that both $v \in \mathfrak{p}_i$ and $v \in \mathfrak{p}_j$.

Clearly, there is an isomorphism of linearly $\mathfrak{p}$-patched structures from $\mathbb{G}$ to $\mathbb{G}'$ iff the instance $\mathbb{I}$ has a solution.

It is useful to consider a single template $\mathbb{T}_{\mathfrak{p}}$ such that any instance $\mathbb{I}$ obtained from two linearly $\mathfrak{p}$-patched structures $\mathbb{G}, \mathbb{G}'$ as above, is over $\mathbb{T}_{\mathfrak{p}}$. The domain of $\mathbb{T}_{\mathfrak{p}}$ consists of isomorphisms $\alpha : \mathfrak{q} \to \mathfrak{q}'$ between graphs isomorphic to $\mathfrak{p}$, whose vertices are atoms. For each quadruple $\mathfrak{q}, \mathfrak{q}', \mathfrak{r}, \mathfrak{r}'$, there is the binary relation $C^{\mathfrak{p},\mathfrak{p}'}_{\mathfrak{r},\mathfrak{r}'}$ defined above. $\mathbb{T}_{\mathfrak{p}}$ is equivariant and, since every such relation is finite, it is locally finite. It is also not difficult to see that this structure is definable.

The next lemma follows from the results in [10].

Lemma 43. *LFP can test isomorphism of linearly $\mathfrak{p}$-patched structures if and only if the template $\mathbb{T}_{\mathfrak{p}}$ has bounded width.*

In [10], the proof of Theorem 41 then proceeds by constructing a finite template which corresponds to $\mathbb{T}_{\mathfrak{p}}$, in an ad-hoc way roughly similar to the one described in Section IV-C, and then further studying its properties. Since the construction of this finite template is technical, its study becomes obfuscated. Instead, using the results from the present paper, we can now work directly with the template $\mathbb{T}_{\mathfrak{p}}$, as sketched below.

By Corollary 35, it can be effectively tested whether $\mathbb{T}_{\mathfrak{p}}$ has bounded width. However, a more effective test is possible, due to the straightforward observation that the template $\mathbb{T}_{\mathfrak{p}}$ has a Maltsev polymorphism, defined by

$$M(\alpha, \beta, \gamma) = \begin{cases} \alpha \cdot \beta^{-1} \cdot \gamma & \text{if } \alpha, \beta, \gamma : \mathfrak{q} \to \mathfrak{q}' \text{ for some} \\ & \quad \mathfrak{q}, \mathfrak{q}' \text{ isomorphic to } \mathfrak{p}, \\ \alpha & \text{otherwise.} \end{cases}$$

The following characterization then follows from Corollary 36.

Lemma 44. *The template $\mathbb{T}_{\mathfrak{p}}$ has bounded width if and only if it has a majority polymorphism.*

By Theorem 19, this last condition can be effectively tested. Lemmas 42, 43, 44 prove Theorem 41.

We believe that definable locally finite templates might arise naturally in other applications to Descriptive Complexity: although any such template is computationally equivalent to a finite one by Theorem 37, the presented reduction heavily relies on the ordering of the universe, and therefore cannot be mimicked by order-invariant logics, such as LFP.

ACKNOWLEDGMENTS

We are grateful to Manuel Bodirsky, Jakub Bulin and Marcin Kozik for their patience in answering our questions about various aspects of CSP theory. We also thank the anonymous reviewers for their thorough and helpful comments.

REFERENCES

[1] P. Jeavons, D. Cohen, and M. Gyssens, "Closure properties of constraints," *J. ACM*, vol. 44, no. 4, pp. 527–548, Jul. 1997.

[2] A. Bulatov and V. Dalmau, "A simple algorithm for mal'tsev constraints," *SIAM Journal on Computing*, vol. 36, no. 1, pp. 16–27, 2006. [Online]. Available: http://dx.doi.org/10.1137/050628957

[3] A. Bulatov, P. Jeavons, and A. Krokhin, "Classifying the complexity of constraints using finite algebras," *SIAM Journal on Computing*, no. 34, pp. 720–742, 2005.

[4] A. Bulatov, A. Krokhin, and P. Jeavons, "Constraint satisfaction problems and finite algebras," in *Proceedings of ICALP'00*, no. 1853, 2000, pp. 272–282, longer version available as an OUCL Technical Report : http://web.comlab.ox.ac.uk/oucl/publications/tr/tr-4-99.html.

[5] L. Barto and M. Kozik, "Absorbing subalgebras, cyclic terms, and the constraint satisfaction problem," *Log. Meth. Comp. Sci.*, vol. 8(1), 2012.

[6] V. Pestov, "On free actions, minimal flows, and a problem by Ellis," *Trans. Amer. Math. Soc.*, vol. 350, pp. 4149–4165, 1998.

[7] A. Kechris, V. Pestov, and S. Todorcevic, "Fraïssé limits, Ramsey theory, and topological dynamics of automorphism groups," *Geometric & Functional Analysis GAFA*, vol. 15, no. 1, pp. 106–189, 2005.

[8] J. Cai, M. Fürer, and N. Immerman, "An optimal lower bound on the number of variables for graph identifications," *Combinatorica*, vol. 12, no. 4, pp. 389–410, 1992.

[9] L. Barto and M. Pinsker, "The basic CSP reductions revisited," announced Nov. 2014. [Online]. Available: http://www.karlin.mff.cuni.cz/~barto/Articles/Banff_Barto.pdf

[10] B. Klin, S. Lasota, J. Ochremiak, and S. Toruńczyk, "Turing machines with atoms, constraint satisfaction problems, and descriptive complexity," in *Procs. of CSL-LICS'14*, 2014, pp. 58:1–58:10.

[11] M. H. Freedman, "K-sat on groups and undecidability," in *Procs. STOC*, ser. STOC '98, 1998, pp. 572–576.

[12] H. Chen, "Periodic constraint satisfaction problems: Tractable subclasses," *Constraints*, vol. 10, no. 2, pp. 97–113, 2005.

[13] S. S. Dantchev and F. D. Valencia, "On the computational limits of infinite satisfaction," in *Procs. (SAC)*, 2005, pp. 393–397.

[14] M. Bodirsky and J. Nesetril, "Constraint satisfaction with countable homogeneous templates," *J. Log. Comput.*, vol. 16, no. 3, pp. 359–373, 2006.

[15] M. Bodirsky, "Cores of countably categorical structures," *Logical Methods in Computer Science*, vol. 3, no. 1, 2007.

[16] M. Bodirsky and M. Pinsker, "Topological birkhoff," *CoRR*, vol. abs/1203.1876, 2012. [Online]. Available: http://arxiv.org/abs/1203.1876

[17] M. Bodirsky and J. Kára, "The complexity of temporal constraint satisfaction problems," *J. ACM*, vol. 57, no. 2, 2010.

[18] M. Bodirsky and M. Pinsker, "Schaefer's theorem for graphs," in *Procs. STOC*, 2011, pp. 655–664.

[19] M. Bodirsky and V. Dalmau, "Datalog and constraint satisfaction with infinite templates," *J. Comput. Syst. Sci.*, vol. 79, no. 1, pp. 79–100, 2013.

[20] M. Bodirsky and M. Pinsker, "Reducts of ramsey structures," *CoRR*, vol. abs/1105.6073, 2011.

[21] A. M. Pitts, *Nominal Sets: Names and Symmetry in Computer Science*. Cambridge University Press, 2013, vol. 57.

[22] M. Bojańczyk and S. Toruńczyk, "Imperative programming in sets with atoms," in *Procs. FSTTCS 2012*, ser. LIPIcs, vol. 18, 2012, pp. 4–15.

[23] M. Bojańczyk, B. Klin, and S. Lasota, "Automata theory in nominal sets," *Log. Meth. Comp. Sci.*, vol. 10, 2014.

[24] M. Bojanczyk, B. Klin, S. Lasota, and S. Torunczyk, "Turing machines with atoms," in *LICS*, 2013, pp. 183–192.

[25] W. Hodges, *Model Theory*, ser. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1993, no. 42.

[26] M. Bojańczyk, L. Braud, B. Klin, and S. Lasota, "Towards nominal computation," in *Procs. POPL 2012*, 2012, pp. 401–412.

[27] M. Bodirsky, M. Pinsker, and T. Tsankov, "Decidability of definability," *The Journal of Symbolic Logic*, vol. 78, no. 04, pp. 1036–1054, 2013.

[28] G. Buntrock, C. Damm, U. Hertrampf, and C. Meinel, "Structure and importance of logspace-mod-classes." in *Procs. STACS*, ser. Lecture Notes in Computer Science, vol. 480, 1991, pp. 360–371.

[29] P. Idziak, P. Markovic, R. McKenzie, M. Valeriote, and R. Willard, "Tractability and learnability arising from algebras with few subpowers," in *Logic in Computer Science, 2007. LICS 2007. 22nd Annual IEEE Symposium on*, July 2007, pp. 213–224.

[30] J. Berman, P. Idziak, P. Markovic, R. McKenzie, M. Valeriote, and R. Willard, "Varieties with few subalgebras of powers," *Transactions of The American Mathematical Society*, vol. 362, pp. 1445–1473, 2009.
[31] B. Larose and L. Zádori, "Bounded width problems and algebras," *Algebra universalis*, vol. 56, no. 3-4, pp. 439–466, 2007.
[32] L. Barto and M. Kozik, "Constraint satisfaction problems solvable by local consistency methods," *J. ACM*, vol. 61, no. 1, pp. 3:1–3:19, Jan. 2014.
[33] M. Kozik, A. Krokhin, M. Valeriote, and R. Willard, "Characterizations of several Maltsev conditions," *Algebra Universalis*, 2015, to appear.
[34] V. Dalmau and B. Larose, "Maltsev + datalog −> symmetric datalog," in *Proceedings of the Twenty-Third Annual IEEE Symposium on Logic in Computer Science (LICS 2008)*. IEEE Computer Society Press, June 2008, pp. 297–306.
[35] N. Immerman, *Descriptive complexity*, ser. Graduate texts in computer science. Springer, 1999.
[36] ——, "Upper and lower bounds for first order expressibility," *J. Comput. Syst. Sci.*, vol. 25, no. 1, pp. 76–98, 1982.
[37] M. Y. Vardi, "The complexity of relational query languages (extended abstract)," in *Procs. STOC*, 1982, pp. 137–146.
[38] C. M. Papadimitriou, *Computational complexity*. Addison-Wesley, 1994.

APPENDIX

A. Proofs from Section II

1) Proof of Theorem 10: The statement (1) follows from the observation that if X is defined by an expression e without free variables, then $\pi \cdot X = X$ by definition. As mentioned, this implies that $\mathrm{Aut}(\mathcal{A})$ acts on X, since $\pi \cdot x \in \pi \cdot X = X$ for $\pi \in \mathrm{Aut}(\mathcal{A}), x \in X$.

In the rest of the proof, the following notion will be useful. The *quantifier-free* type of a valuation $\mathrm{val} : V \to \mathcal{A}$ is the formula

$$\tau_{\mathrm{val}} = \bigwedge_{\substack{v,w \in V: \\ \mathrm{val}(v)=\mathrm{val}(w)}} (v = w) \quad \wedge \quad \bigwedge_{\substack{v,w \in V: \\ \mathrm{val}(v)\neq\mathrm{val}(w)}} (v \neq w).$$

(In general, the quantifier-free type of a valuation $\mathrm{val} : V \to \mathbb{A}$ in a relational structure $\mathbb{A}$ is the conjunction of all literals l with free variables contained in V such that $\mathbb{A}, \mathrm{val} \models l$. Here, $\mathbb{A}$ is the structure $(\mathcal{A}, =)$.)

Note that for a given finite set V, each quantifier-free type is of a polynomially bounded size with respect to $|V|$. In particular, there are finitely many quantifier-free formulas with free variables V, and the set $\{\tau_{\mathrm{val}} \colon \mathrm{val} : V \to \mathcal{A}\}$ is finite and computable.

The statement (2) is an immediate consequence of the following result, which is known as *quantifier-elimination* among model theorists.

Fact 45. *Every first order formula ϕ can be translated in polynomial space into a quantifier-free formula ϕ', such that ϕ and ϕ' are equivalent in the structure $(\mathcal{A}, =)$.*

Proof. The proof proceeds by induction on the size of a first order formula ϕ. If ϕ is quantifier-free, we are done. In the inductive step, it is enough to consider the case when ϕ is of the form $\exists x.\psi$ and ψ is already quantifier-free; the other case when ϕ is a boolean combination of quantifier-free formulas is trivial.

Let V be the set of free variables of the formula ψ. By homogeneity of $(\mathcal{A}, =)$, the formula $\phi = \exists x.\psi$ is equivalent to the formula

$$\phi' = \bigvee_{\tau} \psi_\tau,$$

where τ ranges over all (finitely many) quantifier-free types of valuations $\mathrm{val} : V \to \mathcal{A}$, and ψ_τ is the formula ψ, with every predicate $x = v$ replaced by $\top$ or $\bot$, depending on whether $\tau \models (x = v)$ or not. □

We prove the statement (3). The interesting case is when X is defined by an expression of the form

$$\{e \mid a_1, \ldots, a_n \in \mathcal{A}, \phi\}$$

(no valuation is needed, since by assumption, the entire expression has no free variables).

Let V denote the set of free variables of the expression e, i.e.,

$$V = \{a_1, \ldots, a_n\}.$$

It follows from homogeneity of $(\mathcal{A}, =)$ that if $\mathrm{val}_1, \mathrm{val}_2 : V \to \mathcal{A}$ are two valuations with the same quantifier-free type (i.e., $\tau_{\mathrm{val}_1} = \tau_{\mathrm{val}_2}$), then $e[\mathrm{val}_1]$ and $e[\mathrm{val}_2]$ are in the same orbit with respect to $\mathrm{Aut}(\mathcal{A})$. It follows that each orbit of X is defined by an expression of the form

$$\{e \mid a_1, \ldots, a_n \in \mathcal{A}, \phi \wedge \tau\},$$

where τ is a quantifier-free type of valuations $\mathrm{val} : V \to \mathcal{A}$. Moreover, there are only finitely many orbits.

This yields the third statement, and also the last one, since if Y is equivariant, then it is a union of orbits of X under the action of $\mathrm{Aut}(\mathcal{A})$. □

2) Proof of Proposition 11: Given expressions e, d, one can compute first order formulas $\tau^{\in}_{e,d}$, $\tau^{\subseteq}_{e,d}$ and $\tau^{=}_{e,d}$ such that for any valuation val,

- $\tau^{\in}_{e,d}[\mathrm{val}]$ holds iff $e[\mathrm{val}] \in d[\mathrm{val}]$,
- $\tau^{\subseteq}_{e,d}[\mathrm{val}]$ holds iff $e[\mathrm{val}] \subseteq d[\mathrm{val}]$,
- $\tau^{=}_{e,d}[\mathrm{val}]$ holds iff $e[\mathrm{val}] = d[\mathrm{val}]$.

These formulas are computed by mutual recursion. The most interesting case is where both e and d are set-builder expressions, say:

$$e = \{e' \mid v_1, \ldots, v_n \in \mathcal{A}, \phi\}$$
$$d = \{d' \mid w_1, \ldots, w_m \in \mathcal{A}, \psi\},$$

where we define:

$$\tau^{\in}_{e,d} = \exists w_1. \cdots \exists w_m.(\psi \wedge \tau^{=}_{e,d'})$$
$$\tau^{\subseteq}_{e,d} = \forall v_1. \cdots \forall v_n.(\phi \to \tau^{\in}_{e',d})$$
$$\tau^{=}_{e,d} = \tau^{\subseteq}_{e,d} \wedge \tau^{\subseteq}_{d,e}.$$

By quantifier elimination, these formulas are decidable in polynomial space. □

3) Proof of Lemma 12: Let $X_1, \ldots, X_n$ denote the orbits of X. It follows from the proof of Theorem 10 that the sequence $X_1, \ldots, X_n$ can be effectively computed (possibly with repetitions), and from Proposition 11 it follows that repetitions can be removed. Moreover, each orbit X_i is represented by an expression of the form

$$\{e : a_1, \ldots, a_n \in \mathcal{A}, \tau\},$$

where τ is a satisfiable quantifier-free type. As a representative of the orbit X_i, pick the element $e[\mathrm{val}]$, where $\mathrm{val} : \{a_1, \ldots, a_n\} \to \{\underline{1}, \ldots, \underline{n}\}$ is any assignment satisfying τ. □

B. Proofs from Section III

1) Proof of Lemma 15: Here we do not assume $\mathbb{I}$ or $\mathbb{T}$ to be definable or equivariant.

For any $x \in I$, let $V(x) \subseteq T$ be the set of possible values of x in all solutions of $\mathbb{I}$. Since $\mathbb{I}$ is constrained and $\mathbb{T}$ is locally finite, every $V(x)$ is finite.

The set of functions:

$$F(\mathbb{I}, \mathbb{T}) = \{f : I \to T \mid f(x) \in V(x) \text{ for } x \in I\}$$

equipped with the product topology inherited from T^I, is a topological product of discrete finite spaces:

$$F(\mathbb{I},\mathbb{T}) = \prod_{x\in\mathbb{I}} V(x)$$

therefore (by Tychonoff's theorem) it is compact. Note that $\hom(\mathbb{I},\mathbb{T}) \subseteq F(\mathbb{I},\mathbb{T})$.

Consider any $f \in F(\mathbb{I},\mathbb{T})$ that is *not* a solution of $\mathbb{I}$. This means that there is a constraint $((x_1,\dots,x_n),R)$ in $\mathbb{I}$ such that $(f(x_1),\dots,f(x_n)) \notin R$. Then, for $J = \{x_1,\dots,x_n\}$, we have

$$\mathcal{B}_J(f) \cap \hom(\mathbb{I},\mathbb{T}) = \emptyset.$$

As a result, $\hom(\mathbb{I},\mathbb{T})$ is a closed subspace of the compact space $F(\mathbb{I},\mathbb{T})$, hence it is itself compact. □

2) Proof of Lemma 16: Atom permutations $\pi \in \mathrm{Aut}(\mathcal{A})$ act on functions $f : I \to T$ by:

$$(\pi \cdot f)(x) = \pi \cdot (f(\pi^{-1} \cdot x)).$$

We want to show that if f is a solution of $\mathbb{I}$ then so is $\pi \cdot f$. To this end, consider any constraint $(\bar{x}, R)$ in $\mathbb{I}$, with $\bar{x}$ of length n. Since $\mathbb{I}$ is equivariant,

$$(\pi^{-1} \cdot \bar{x}, \pi^{-1} \cdot R)$$

is also a constraint in $\mathbb{I}$. Then, since f is a solution of $\mathbb{I}$:

$$f^n(\pi^{-1} \cdot \bar{x}) \in \pi^{-1} \cdot R$$

and finally:

$$(\pi \cdot f^n)(\bar{x}) \in R.$$

This proves that $\mathrm{Aut}(\mathcal{A})$ acts on $\hom(\mathbb{I},\mathbb{T})$. We will now check that this action is continuous.

To this end, consider any solution f of $\mathbb{I}$ and its basic open neighbourhood $\mathcal{B}_J(f)$ as in (1). The inverse image of $\mathcal{B}_J(f)$ along the action is:

$$\overleftarrow{\mathcal{B}_J(f)} = \{(\pi, g) \mid \forall x \in J.\ \pi \cdot f(x) = g(\pi \cdot x)\}$$

and we need to show that it is open in $\mathrm{Aut}(\mathcal{A}) \times \hom(\mathbb{I},\mathbb{T})$.

To this end, recall that I is defined by a set expression. This means that each variable $x \in I$ is also defined by an expression e with a valuation val; let $S_x \subseteq_{\text{fin}} \mathcal{A}$ be the range of val. Note that S_x supports x, i.e., for any atom permutation π, the value $\pi \cdot x$ depends only on how π acts on S_x.

Moreover, since $\mathbb{I}$ is constrained and f is a solution, the element $f(x) \in T$ appears in the definition of $\mathbb{I}$, therefore it is also definable, hence supported by some finite $S_{f(x)} \subseteq \mathcal{A}$.

Now pick any $(\pi, g) \in \overleftarrow{\mathcal{B}_J(f)}$ and define:

$$\begin{aligned} S &= \bigcup_{x\in J} S_x \cup S_{f(x)} \subseteq_{\text{fin}} \mathcal{A} \\ K &= \{\pi \cdot x \mid x \in J\} \subseteq_{\text{fin}} I. \end{aligned}$$

Then, for any $\sigma \in \mathcal{B}_S(\pi)$ and $h \in \mathcal{B}_K(g)$, we have:

$$\sigma \cdot f(x) = \pi \cdot f(x) = g(\pi \cdot x) = h(\pi \cdot x) = h(\sigma \cdot x)$$

therefore $(\sigma, h) \in \overleftarrow{\mathcal{B}_J(f)}$. As a result:

$$\mathcal{B}_S(\pi) \times \mathcal{B}_K(g) \subseteq \overleftarrow{\mathcal{B}_J(f)}$$

therefore $\overleftarrow{\mathcal{B}_J(f)}$ is indeed open and the action is continuous. □

3) Proof of Lemma 18: Assume that $\mathbb{I}$ is defined by a set expression e as explained in Section II-A. By Lemma 12 it is possible to compute a system of representatives of the orbits of I with respect to monotone atom permutations. Note that every monotone-equivariant function from I to T is fully determined by its values on the representatives.

Since $\mathbb{T}$ is locally finite and $\mathbb{I}$ is constrained, for any variable $x \in I$ there are only finitely many elements in T that can be values of x in a solution of $\mathbb{I}$. Moreover, all these elements can be effectively enumerated, given a definiton of $\mathbb{I}$.

Altogether, this means that there are finitely many candidates for monotone-equivariant solutions of $\mathbb{I}$ and that they can be effectively enumerated.

Finally, given a monotone-equivariant function $f : I \to T$, it can be effectively checked whether it is a solution of $\mathbb{I}$. To this end, one computes representatives of the orbits of constraints in $\mathbb{I}$; it is enough to check that all representative constraints are satisfied, and for a given constraint it is straightforward to check whether f satisfies it. □

4) Proof of Theorem 22: For simplicity, we will assume that all languages are over $\mathbb{B} = \{0,1\}$, and all complexity classes concern languages over $\mathbb{B}$.

Let $L \in \exp(\mathcal{C})$. We have to show how to reduce L to CSP-Inf($\mathbb{T}$). By the definition of $\exp(\mathcal{C})$, there is a natural number k such that $\mathrm{pad}_k(L) \in \mathcal{C}$. Since CSP($\mathbb{T}$) is $\mathcal{C}$-hard, there is a logspace (for the proof to work it is enough to assume polylogspace) Turing machine M which, given a word $w \in \mathbb{B}^*$, produces an instance $I(w)$ of CSP($\mathbb{T}$) such that $I(w)$ is satisfiable iff $w \in \mathrm{pad}_k(L)$. Without loss of generality, we can assume that each variable in $I(w)$ is indexed by $p(n)$ bits, where p is a fixed polynomial, and n is the logarithm of the length of w (so that positions in w can be addressed by elements of $\mathbb{B}^n$).

We can assume that M, instead of producing an instance, receives an input $(i, b^1, \dots, b^{r_i}, w)$, where r_i is the arity of R_i and $b^1,\dots,b^{r_i}$ are words of length $p(n)$, and accepts iff the instance $I(w)$ includes the constraint $\big((x_{b^1},\dots,x_{b^{r_i}}), R_i\big)$. Furthermore, we can also change M so that w is not given as an input, but via an oracle. The machine M has a question tape where it can write an address $j \in \mathbb{B}^n$, and the oracle answers whether $w_j = 1$. Such a machine M runs in space polynomial in its input (not counting the oracle).

Recall that a classical PSPACE-complete problem is QBF [38]. We can encode the working of M on words of length n using a QBF formula ϕ of length polynomial in n, which additionally has an access to i, all the bits of $b^1,\dots,b^{r_i}$, and an additional logical operation $\mathrm{Input}(x_1,\dots,x_n)$ which is true iff w contains 1 at the position addressed by the sequence of bits $(x_1,\dots,x_n)$. The instance $I(w)$ can be defined in terms of this formula thus:

$$\big\{\big((x_{b^1},\dots,x_{b^{r_i}}), R_i\big) : \phi_{[w]}(b^1,\dots,b^{r_i})\big\},$$

where $\phi_{[w]}$ is the formula ϕ where every occurrence of $\mathrm{Input}(x_1,\dots,x_n)$ is resolved to 0 or 1 according to w. In

the next paragraph we explain how the QBF formula ϕ is generated.

We can transform M running in PSPACE into one running in alternating polynomial time (the usual proof of PSPACE = AP proceeds with no change in the presence of an oracle), and then transform the alternating time machine running in time $t(n)$ and space $s(n)$ (both polynomial) into a formula by listing all the $t(n)$ configurations of length $s(n)$, using universal and existential quantifiers for alternations, and binary connectives to check whether the run is correct.

Now, let $v \in \mathbb{B}^m$. Let $n = m^k$ be the logarithm of the length of $\mathrm{pad}_k(v)$. We have to construct an instance $I^*(v)$ of CSP-Inf($\mathbb{T}$) such that $v \in L$ iff $I^*(v)$ is satisfiable. We construct $I^*(v)$ as follows: the variables are indexed by $(p(n)+1)$-tuples of atoms (formally, they are $(p(n)+1)$-tuples of atoms), and for each relation $R_i \in \mathbb{T}$, we take the constraints

$$\left\{\left((x_{a,a_1^1\ldots a_{p(n)}^1},\ldots,x_{a,a_1^{r_i},\ldots a_{p(n)}^{r_i}}),R_i\right) : [\phi](a,a_1^1,\ldots,a_{p(n)}^{r_i})\right\}$$

where the first order formula with equality $[\phi]$ is obtained from ϕ and w recursively in the following way:

- $[b_j^i] = (a = a_j^i)$
- $[\exists x\psi] = \exists a_x[\psi]$, $[\forall x\psi] = \forall a_x[\psi]$
- $[x] = (a = a_x)$
- $[\psi_1 \vee \psi_2] = [\psi_1] \vee [\psi_2]$, $[\psi_1 \wedge \psi_2] = [\psi_1] \wedge [\psi_2]$, $[\neg\psi_1] = \neg[\psi_1]$
- $[\mathrm{Input}(x_1,\ldots,x_n)]$ is the formula $\rho(a,a_{x_1},\ldots,a_{x_n})$ which is true iff the $(b_1\ldots b_n)$-th symbol of $w = \mathrm{pad}_k(v)$ is 1, where $b_i = 1$ iff $a_{x_i} = a$. It is straightforward to construct such ρ of size polynomial in m; indeed, we just have to take a disjunction of m clauses of form

$$b_1 = j_1 \wedge b_2 = j_2 \wedge \ldots \wedge b_{\log m} = j_{\log m}$$

for the $(j_1,\ldots,j_{\log m})$ such that $w_j = 1$.

The following conditions are then equivalent:

- $v \in L$,
- $\mathrm{pad}_k(v) \in \mathrm{pad}_k(L)$,
- $I(\mathrm{pad}_k(v))$ is satisfiable,
- $I^*(v)$ is satisfiable,

which completes the proof. □

C. Proofs from Section IV

1) Proof of Lemma 24: The equivalence (1↔2) is standard, so we omit it. The implication (2→3) is obvious, so it remains to show the opposite implication.

Let $R = \Phi(\mathbb{T})$ for some generalized pp-formula $\Phi = (\mathbb{I},\alpha)$, with some $\alpha : X \to \mathbb{I}$, where $X = \{x_1,\ldots,x_n\}$ is the set of free variables. Assume that for each $x \in X$, the element $\alpha(x)$ is not isolated in $\mathbb{I}$ (i.e., appears in some constraint), as the general case reduces to this case easily.

Let $c_0, c_1, \ldots,$ be a sequence of all the constraints in $\mathbb{I}$, and let $\mathbb{I}_n$ be the subinstance of $\mathbb{I}$ induced by the constraints c_i with $i < n$, and containing the variables $\alpha(x)$, for $x \in X$ i.e.,

- the variables of $\mathbb{I}_n$ are the variables which appear in any of the constraints $c_0, c_1, \ldots, c_{n-1}$, together with all variables $\alpha(x)$ for $x \in X$,
- the constraints of $\mathbb{I}_n$ are $c_0,\ldots,c_{n-1}$.

Let Φ_n be the generalized pp-formula $(\mathbb{I}_n,\alpha)$, for $i = 1,2,\ldots$.

By assumption that for each $x \in X$ the variable $\alpha(x)$ is not isolated, and since X is finite, there is a number m_0 such that each $\alpha(x)$ appears in a constraint c_i with $i < m_0$.

By local finiteness of $\mathbb{T}$, it follows that the relation $\Phi_{m_0}(\mathbb{T})$ is finite. Therefore, the sequence $\Phi_m(\mathbb{T})$, for $m > m_0$, is a descending sequence of finite relations, so it stabilizes at some point m_1, i.e., $\Phi_m(\mathbb{T}) = \Phi_{m_1}(\mathbb{T})$ for $m > m_1$.

The following equation

$$\Phi(\mathbb{T}) = \bigcap_{n=1}^{\infty} \Phi_n(\mathbb{T}) \tag{2}$$

then implies that $R = \Phi(\mathbb{T}) = \Phi_{m_1}(\mathbb{T})$, so R is defined by a finite generalized pp-definition.

It remains to prove equation (2). The left-to-right inclusion is clear, since every homomorphism from $\mathbb{I}$ to $\mathbb{T}$ induces a homomorphism from $\mathbb{I}_n$ to $\mathbb{T}$. For the right-to-left inclusion, suppose that $(t_1,\ldots,t_n) \in \Phi_n(\mathbb{T})$ for all n. Let $f_0 : \mathbb{I}_0 \to \mathbb{T}$ be defined by $f_0(\alpha(x_i)) = t_i$ for $i = 1,\ldots,n$. Then for each n there is a homomorphism $f_n : \mathbb{I}_n \to \mathbb{T}$ extending $f_0 : \mathbb{I}_0 \to \mathbb{T}$. Applying Lemma 25 yields a homomorphism $f : \mathbb{I} \to \mathbb{T}$ extending f_0, which is a witness of $(t_1,\ldots,t_n) \in \Phi(\mathbb{T})$. □

2) Proof of Lemma 25: We show the right-to-left implication, as the other one is obvious. We assume that the structure $\mathbb{I}$ is constrained, i.e., it does not have isolated nodes, and that the substructures $\mathbb{I}_n$ are finite (the general case reduces to this case easily).

Take any element $x \in I$. Since $\mathbb{I}$ is constrained, x appears on the j-th coordinate of some tuple t constrained to some relation R from $\mathbb{T}$. Let U_x be the projection of R on the j-th coordinate. It is not difficult to see that if for every $n \geq 0$ there exists a homomorphism from $\mathbb{I}_n$ to $\mathbb{T}$ then for every $n \geq 0$ there exists one that, for every $x \in I_n$, maps x to an element of U_x.

Consider the compact subspace $\prod_{x\in I} U_x$ of T^I. For every $\mathbb{I}_n$ let S_n be the set of all those mappings $f \in \prod_{x\in I} U_x$ for which $f|_{I_n}$ is a homomorphisms from $\mathbb{I}_n$ to $\mathbb{T}$. Every S_i is a nonempty, closed subset of $\prod_{x\in\mathbb{I}} U_x$. Therefore, the descending sequence $S_1 \supseteq S_2 \supseteq \ldots$ has a nonempty intersection. Since $\bigcup_n \mathbb{I}_n = \mathbb{I}$, every mapping $f : \mathbb{I} \to \mathbb{T}$ which belongs to this intersection is a homomorphism from $\mathbb{I}$ to $\mathbb{T}$. □

3) Proof of Proposition 26: Let $\mathcal{R}$ be the family of relations of $\mathbb{B}$. Since the set $\mathcal{R}$ is definable, it is orbit-finite with respect to the action of monotone permutations. Let $R_1,\ldots,R_n$ be representatives of the orbits of $\mathcal{R}$. By assumption, $R_1,\ldots,R_n \in \mathrm{ppDef}\,\mathbb{C}$. For $i = 1,\ldots,n$, let $\Phi_i = (\mathbb{I}_i,\alpha_i)$ be a finite generalized pp-formula such that $\Phi_i(\mathbb{C}) = R_i$.

We may assume that the domains of $\mathbb{I}_i$ are pairwise disjoint, and that their elements are indexed by integers. Since each constraint in $\mathbb{I}_i$ uses a finite relation in $\mathbb{C}$, these constraints are definable sets with atoms, and, in effect, Φ_i is a definable set with atoms.

We now describe how to convert a finite instance $\mathbb{I} = (I, C)$ over $\mathbb{B}$ to an instance $\mathbb{I}' = (I', C')$ over $\mathbb{C}$.

For each constraint $c = ((x_1, \ldots, x_n), R)$ in $\mathbb{I}$, choose R_i which is in the same orbit as R, and any monotone-permutation π_c which maps R_i to R (such a permutation can be computed as a piecewise-linear monotone bijection of the rational numbers). Apply π_c to the finite generalized pp-formula Φ_i, yielding a finite generalized pp-formula

$$\Phi_c = \pi_c \cdot \Phi_i$$

(recall that Φ_i is a definable set with atoms). By equivariance of $\mathbb{C}$,

$$\Phi_c(\mathbb{C}) = (\pi_c \cdot \Phi_i)(\mathbb{C}) = \pi_c \cdot \Phi_i(\mathbb{C}) = \pi_c \cdot R_i = R.$$

To summarize, for each constraint $c = ((x_1, \ldots, x_n), R)$ of $\mathbb{I}$, there is a finite generalized pp-formula $\Phi_c = (\mathbb{I}_c, \alpha_c)$ such that $R = \Phi_c(\mathbb{C})$. Moreover, the pp-formula Φ_c can be computed in polynomial time from c.

The rest of the construction is standard. We define the structure $\mathbb{J}$ as the disjoint union

$$\mathbb{J} = I \cup \coprod_{c \in C} \mathbb{I}_c,$$

where I is treated as a set with no relations. Define $\mathbb{I}'$ as the quotient $\mathbb{J}/\sim$, where $\sim$ is the smallest equivalence relation such that $v \sim x$ iff $v \in I$ and there is a constraint $c = ((x_1, \ldots, x_n), R)$ in $\mathbb{I}$ and $i \in \{1, \ldots, n\}$ where

- $v = x_i$, and
- x is the i-th free variable of the generalized pp-formula Φ_c.

The structure $\mathbb{I}'$ can be computed in polynomial time from $\mathbb{I}$. It is easy to check that $\mathbb{I}$ maps to $\mathbb{B}$ if and only if $\mathbb{I}'$ maps to $\mathbb{C}$. □

4) Proof of Proposition 27: In this abstract proof, it will be convenient to use the following conventions. If X is a set and $f : A \to B$ is a function then by $f^X : A^X \to B^X$ we denote the function defined by $f^X(\alpha) = \alpha; f$. For each $x \in X$ the projection $\pi_x : A^X \to A$ is defined by $\pi_x(f) = f(x)$. In the case when $X = \{1, \ldots, n\}$ for some natural number then A^X and f^X are denoted by A^n and f^n, respectively. For a function $f : Y \to A^X$, by $f^\flat : X \to A^Y$ we denote its *transpose*, defined by $f^\flat(x)(y) = f(y)(x)$ for $x \in X, y \in Y$.

In this proof, relations and functions have set-arities which are finite sets, rather than natural numbers. Formally, if A is a set and X is a finite set, then a *relation with set-arity* X on A is a subset of A^X, and a *function* (or *operation*) *with set-arity* X on A is a function $f : A^X \to A$.

If n is a natural number, then identifying A^n with $A^{\{1,\ldots,n\}}$ allows to interpret relations of arity n on A as relations with set-arity $\{1, \ldots, n\}$ (and similarly for operations). Conversely, to view relations with set-arities as normal relations, fix for every finite set X a bijection from X to $\{1, \ldots, n\}$ where $n = |X|$, thus identifying A^X with A^n.

We now proceed to the proof of Proposition 27. Let Y be the set-arity of R, i.e. $R \subseteq A^Y$.

For the left-to-right implication, let $\Phi = (\mathbb{I}, \alpha)$ be a generalized pp-formula with variables Y such that $R = \Phi(\mathbb{T})$. We show that R is invariant under every polymorphism of $\mathbb{T}$. Let $g : \mathbb{T}^X \to \mathbb{T}$ be a polymorphism. Consider any X-tuple of tuples in R, i.e., a function $u : X \to R \subseteq T^Y$. Let $u^\flat : Y \to T^X$ be the transpose of u. We need to show that the composition $(u^\flat; g) : Y \to T$ belongs to $R = \Phi(\mathbb{T})$, i.e. is of the form $\alpha; k$ for some homomorphism $k : \mathbb{I} \to \mathbb{T}$.

For every $x \in X$, $u(x)$ is a tuple in $R = \Phi(\mathbb{T})$, i.e. is of the form $\alpha; h$ for some homomorphism $h \in \hom(\mathbb{I}, \mathbb{T})$. Using the axiom of finite choice, there is a function $f : X \to \hom(\mathbb{I}, \mathbb{T})$ such that $u(x) = \alpha; f(x)$ for all $x \in X$. Its transpose is a homomorphism $f^\flat : \mathbb{I} \to \mathbb{T}^X$ such that $u^\flat = \alpha; f^\flat$. The composition $k = (f^\flat; g) : \mathbb{I} \to \mathbb{T}$ is again a homomorphism, so $u^\flat; g = \alpha; k$ indeed belongs to $\Phi(\mathbb{T}) = R$.

For the right-to-left implication, suppose that R is invariant under the polymorphisms of $\mathbb{T}$. We show that

$$R = \{i^\flat; f \mid f : \mathbb{T}^R \to \mathbb{T} \text{ is a polymorphism}\},$$

where $i : R \to T^Y$ is the inclusion mapping. The above equation immediately implies that R is definable by the generalized pp-formula $\Psi = (\mathbb{T}^R, i^\flat)$. Lemma 24 then implies that R is pp-definable. It therefore remains to prove the above equation.

For the right-to-left inclusion, observe that if $f : \mathbb{T}^R \to \mathbb{T}$ is a polymorphism, then by our assumption f preserves R, so in particular $(i^\flat; f) \in R$. The left-to-right inclusion is also clear, since for any $r \in R$, we have $r = i^\flat; \pi_r$ where $\pi_r : \mathbb{T}^R \to \mathbb{T}$ is the projection polymorphism defined by $\pi_r(t) = t(r)$. □

5) Proof of Proposition 30: We separately show the equivalences (1↔3) and (1↔2).

(1→3). Let $f : \mathbb{T}|_A \to \mathbb{T}|_A$ be an endomorphism; we show that it is mono. It suffices to show that f extends to an endomorphism $\hat{f}$ of $\mathbb{T}$. Indeed, by the the assumption (1), the mapping $\hat{f}$ is a monomorphism, so its restriction f must be mono as well.

Consider the generalized pp-definition $\Phi = (\mathbb{T}, \alpha)$, where $\alpha : A \to \mathbb{T}$ is the inclusion. Let $R = \Phi(\mathbb{T}) \subseteq T^A$. The relation R is finite, as there are no isolated nodes in $\mathbb{T}$, so $R \in \mathrm{ppDef}(\mathbb{T})$, by Lemma 24. By definition of $\mathbb{T}|_A$, the restriction $R|_A$ of R to A is a relation of $\mathbb{T}|_A$, and therefore is preserved by f. Clearly, the identity mapping id_A belongs to $R|_A$, and since f preserves $R|_A$, it follows that $f = f; id_A \in R|_A$. This means that f extends to an endomorphism $\hat{f}$ of $\mathbb{T}$. This proves the implication (1→3).

(3→1). Let f be an endomorphism of $\mathbb{T}$, and let x, y be two distinct elements of $\mathbb{T}$. We show that $f(x) \neq f(y)$. Since $\mathbb{T}$ has no isolated vertices, there are finite, pp-definable unary predicates $U, V \in \mathrm{ppDef}\,\mathbb{T}$ such that $x \in U$ and $y \in V$. Then $\mathbb{T}|_{U \cup V}$ is a core by (3), and f induces its endomorphism; in particular $f(x) \neq f(y)$.

We now proceed to the proof of the equivalence (1↔2). The left-to-right implication is immediate. To prove other the implication, assume that f is an endomorphism which is not mono. We prove that there exists a monotone-equivariant

endomorphism h which is not mono. The proof invokes Pestov's theorem, as described below.

Let $x, y \in \mathbb{T}$ be distinct elements such that $f(x) = f(y)$. By assumption that $\mathbb{T}$ does not have isolated nodes, there is a (finite) relation R of $\mathbb{T}$, such that x appears on some coordinate of some tuple in R. By taking the appropriate projection of R, we obtain a finite unary relation $U \in \mathrm{ppDef}\,\mathbb{T}$ which contains x. Similarly, there is finite, unary relation $V \in \mathrm{ppDef}\,\mathbb{T}$ which contains y. We claim that, without loss of generality, we may assume that $U = V$.

Indeed, if $x \in V$, then we can simply replace U by V. Assume now that $x \notin V$. Since f is an endomorphism, $f(x) \in U$ and $f(y) \in V$, so $f(x) = f(y) \in U \cap V$. Because $U \cap V$ is finite and preserved by f, there is a finite number n such that f^n is idempotent on $U \cap V$, i.e., $f^{2n}(v) = f^n(v)$ for $v \in U \cap V$. Take $y' = f^n(x)$, we have:

- $y' \in U \cap V \subseteq U$ because $f(x) \in U \cap V$ and f preserves $U \cap V$
- $x \neq y'$ because $x \notin U \cap V$,
- $f^n(x) = f^n(y')$ because f^n is idempotent on $U \cap V$.

Replacing f by f^n, y by y' and V by U, we get that $f(x) = f(y)$, $x \neq y$ and $x, y \in U$, where U is a finite, unary relation in $\mathrm{ppDef}\,\mathbb{T}$.

Let $G = \mathrm{Aut}(\mathcal{A}, <)$ denote the group of monotone atom permutations. By equivariance of $\mathbb{T}$ under the action of G, the orbit of U, i.e.,

$$G \cdot U \stackrel{\text{def}}{=} \{\pi \cdot U \mid \pi \in G\},$$

is a family of unary predicates which is contained in $\mathrm{ppDef}\,\mathbb{T}$.

Claim 46. *There is an endomorphism g whose restriction to V is not mono, for all $V \in G \cdot U$.*

Let us choose a sequence $\pi_1, \pi_2, \ldots \in G$ of monotone permutations such that $G \cdot U = \{\pi_n \cdot U : n = 1, 2, \ldots\}$. For $n = 1, 2, \ldots$, let $U_n = \pi_n \cdot U$. We define f_n inductively for $n = 0, 1, 2, \ldots$, so that f_n is an endomorphism of $\mathbb{T}$ whose restriction to U_i is not mono, for $i = 1, 2, \ldots, n$. Let $f_0 = f$. Assuming that f_n is already defined, f_{n+1} is defined as the composition $(\pi_{n+1} \cdot f); f_n$. Then f_{n+1} satisfies the required property.

By Lemma 15, the sequence of mappings $f_0, f_1, f_2, \ldots$ contains a convergent subsequence, whose limit is an endomorphism g whose restriction to any unary predicate $V \in G \cdot U$ is not mono. This proves the claim.

Claim 47. *There is a monotone-equivariant endomorphism h which is not mono.*

Consider the G-orbit $G \cdot g$ of the mapping g. Note that all mappings in this set are not mono when restricted to U. The topological closure $\overline{G \cdot g}$ of the orbit $G \cdot g$ is a monotone-equivariant, closed set of endomorphisms, and all mappings in this set are also not mono when restricted to U – this follows from the fact that not being mono when restricted to U is a closed property, as U is finite. By Pestov's theorem, there is a monotone-equivariant endomorphism $h \in \overline{G \cdot g}$. This mapping is not mono, since its restriction to U is not mono.

Reassuming, if there exists an endomorphism f of $\mathbb{T}$ which is not mono, then there exists also a monotone-equivariant one. This finishes the proof of the implication (2→1). of Proposition 30. □

Turing Machines with Atoms, Constraint Satisfaction Problems, and Descriptive Complexity

Bartek Klin * Sławomir Lasota* Joanna Ochremiak † Szymon Toruńczyk*

University of Warsaw

{klin,sl,ochremiak,szymtor}@mimuw.edu.pl

Abstract

We study deterministic computability over sets with atoms. We characterize those alphabets for which Turing machines with atoms determinize. To this end, the determinization problem is expressed as a Constraint Satisfaction Problem, and a characterization is obtained from deep results in CSP theory. As an application to Descriptive Complexity Theory, within a substantial class of relational structures including Cai-Fürer-Immerman graphs, we precisely characterize those subclasses where the logic IFP+C captures order-invariant polynomial time computation.

Categories and Subject Descriptors F.1.1 [*Models of Computation*]: Turing machines; F.4.1 [*Mathematical Logic*]: Logic and constraint programming; F.2.2 [*Nonnumerical Algorithms and Problems*]: Computations on discrete structures

Keywords Sets with atoms, Turing machines, Constraint Satisfaction Problems, Descriptive Complexity Theory

1. Introduction

Imagine Turing machines which can manipulate not only binary digits, but also *atoms* which come from an infinite, countable set. Moreover, input letters can be finite structures built of atoms. Typical letters include ordered quadruples of atoms, unordered sets of eight atoms, or graphs with ten atoms as nodes. Such machines are called *Turing machines with atoms* [4], or *TMAs* for short.

A TMA is allowed to read and write letters on a tape and store them as parts of its internal state, but it is required to be invariant with respect to bijective atom renaming. For example, if a machine in a state that stores a set of two atoms $\{a, b\}$, upon reading a letter $\{b, c\}$ produces the letter $\{a, c\}$, then in a similar state storing some other set $\{d, e\}$, upon reading $\{e, f\}$, it must produce $\{d, f\}$. Intuitively, atoms have no discernible structure except equality, and a machine may base its actions on comparisons between atoms, but not on the identity of particular atoms. It follows that the language accepted by a TMA is always closed under bijective atom renaming. For example, over the alphabet of unordered pairs of atoms, there is a deterministic TMA recognizing the language of those words which, understood as lists of edges of an undirected graph, describe connected graphs.

In contrast to classical Turing machines, some TMAs do not determinize. In [4], a certain language, over a particular alphabet with each letter built of six atoms, was proved to be recognizable by a nondeterministic TMA (in polynomial time), but not recognizable by any deterministic one. An alphabet for which such a language exists is called *nonstandard*. On the other hand, alphabets such as tuples or finite sets of atoms, are standard: every TMA over them does determinize.

This raises a few questions: how to check whether an alphabet is standard? Is it a decidable property of alphabets? Is the six-atom alphabet of [4] the simplest nonstandard one? In this paper we tackle these questions, and in the process we reveal connections of computation with atoms to some well-studied areas of Theoretical Computer Science, in particular to the theory of Constraint Satisfaction Problems (CSP) [8].

First, we recall from [4] that an alphabet A is standard if and only if a deterministic TMA, given words v and w over A, can decide whether v can be obtained from w by a bijective atom renaming. Then we show that if any deterministic TMA decides this problem then it may be decided by a specific algorithm akin to consistency algorithms studied in CSP theory.

Exploring this connection further, we show how to encode a given alphabet as a finite relational template in the sense of CSP, so that the alphabet is standard if and only if the template is "easy" in a certain well-known sense (specifically, if it admits a majority polymorphism). The latter property is clearly decidable, which gives an effective characterization of standard alphabets. As a direct application, we show that all alphabets with letters built of up to five atoms are standard, so the nonstandard alphabet of [4] is indeed a minimal one. (However, other six-atom nonstandard alphabets do exist.)

The nonstandard alphabet and language defined in [4] resemble the well-known CFI graphs, introduced in [6] to show a limited power of a logic IFP+C over unordered structures (more precisely, an equivalent logic LFP+C was used). We explain this connection by showing that for an alphabet A with atoms, any word defines a relational structure, and over the class of structures obtained in this way, the logic IFP+C captures exactly polynomial time computations by deterministic TMAs. Our results yield a characterization of those classes of structures obtained from words with atoms, over which IFP+C captures polynomial time computations, in the

* Supported by ERC Starting Grant "Sosna".

† Supported by the Polish National Science Centre (NCN) grant 2012/07/B/ST6/01497.

CSL-LICS 2014, July 14–18, 2014, Vienna, Austria.

http://dx.doi.org/10.1145/2603088.2603135

sense used in Descriptive Complexity. This result can be seen as a common generalization of the Immerman-Vardi and the Cai-Fürer-Immerman theorems. In Section 7 we discuss relations with a paper of Atserias, Bulatov and Dawar [1] and also with the graph isomorphism problem for bounded color classes.

2. Turing Machines with Atoms

We begin by recalling some basic notions of sets with atoms [3, 4], also known as Fraenkel-Mostowski sets [10] or nominal sets [14].

Sets with atoms. Fix a countably infinite set $\mathbb{A}$ of *atoms*. A *set with atoms* is any set that can contain atoms or other sets with atoms, in a well-founded way. Formally, sets with atoms are defined by ordinal induction: the empty set is the only set at level 0, and sets at level α either are atoms (which contain no elements) or contain sets at levels smaller than α.

Examples of sets with atoms include:

- any classical set without atoms,
- the set $\mathbb{A}$ itself,
- for any $n \in \mathbb{N}$, the set $\mathbb{A}^n$ of all n-tuples and $\mathbb{A}^{(n)}$ of non-repeating n-tuples of atoms (tuples may be encoded by usual set-theoretic constructions),
- the set $\binom{\mathbb{A}}{n}$ of sets of atoms of size n,
- the set $\mathbb{A}^*$ of finite words over $\mathbb{A}$,
- the set $\mathcal{P}_{\text{fin}}\mathbb{A}$ of all finite subsets of $\mathbb{A}$, etc.

Bijective atom renaming acts on sets with atoms in a canonical way; for instance, it acts coordinatewise on $\mathbb{A}^{(n)}$. For a set with atoms X and a bijection $\pi : \mathbb{A} \to \mathbb{A}$, by $\pi(X)$ we denote the set obtained by consistently replacing atoms in X and in its elements according to π (formally, this is again defined by ordinal induction). We say that a set $S \subseteq \mathbb{A}$ *supports* X if $X = \pi(X)$ for every π which is the identity on S. For example, a tuple $(a, b, c) \in \mathbb{A}^3$ is supported by the set $\{a, b, c\} \subseteq \mathbb{A}$, but also by any larger set.

A set with atoms is *hereditarily finitely supported* if it has some finite support, each of its elements has some finite support, and so on recursively. In this paper we only consider hereditarily finitely supported sets, and so in the following we omit this qualification.

It is not difficult to prove (see e.g. [10, Proposition 3.4]) that every set with atoms has the least finite support with respect to inclusion. By *the* support of a set with atoms X we will mean the least finite support; we denote it by $\sup(X)$. If X is a finite set of atoms, a finite set of such sets or so on, then $\sup(X)$ is the set of those atoms that appear in X.

Equivariance. A set with atoms is *equivariant* if its support is empty (note that its elements need not have the empty support). The example sets listed above are all equivariant.

A relation $R \subseteq X \times Y$ between two equivariant sets with atoms can be seen as a set with atoms itself. It is equivariant if and only if it is closed under the action of atom renaming, i.e., if for any $x \in X$ and $y \in Y$,

$$(x, y) \in R \text{ implies } (\pi(x), \pi(y)) \in R$$

for any bijection $\pi : \mathbb{A} \to \mathbb{A}$. If the relation is (the graph of) a function $f : X \to Y$, this translates to:

$$f(\pi(x)) = \pi(f(x))$$

for any bijection $\pi : \mathbb{A} \to \mathbb{A}$ and any $x \in X$; i.e., equivariant functions are those that commute with atom renaming. It follows that for any $x \in X$ the support of $f(x)$ is contained in the support of x, since a permutation π which fixes x will also fix $f(x)$. The notion of an equivariant function formalizes the intuition of a function that only cares about atom equality, and does not depend on any other structure of the atoms.

For example, the only equivariant function from $\mathbb{A}$ to $\mathbb{A}$ is the identity, the only equvariant functions from $\mathbb{A}^{(n)}$ to $\mathbb{A}$ are the n projections, and the only equivariant function from $\mathbb{A}$ to $\mathbb{A}^n$ is the diagonal. There is no equivariant function from $\binom{\mathbb{A}}{2}$ to $\mathbb{A}$. Indeed, suppose that $f : \binom{\mathbb{A}}{2} \to \mathbb{A}$ is such that, say,

$$f(\{a, b\}) = a$$

for some $a \neq b \in \mathbb{A}$. Then f is not equivariant, since for a bijection π that swaps a and b:

$$f(\pi(\{a, b\})) = f(\{a, b\}) = a \neq b = \pi(a) = \pi(f(\{a, b\})).$$

Intuitively, there is no way of choosing one atom out of two when all one has is atom equality. However,

$$\big\{(\{a, b\}, a) \mid a, b \in \mathbb{A},\ a \neq b\big\}$$

is an equivariant *relation* between $\binom{\mathbb{A}}{2}$ and $\mathbb{A}$. Note that it relates $\{a, b\}$ both to a and b.

Notational convention. Most often we will make a pragmatic distinction between those sets that we consider as collections of interesting elements (for example, the sets $\mathbb{A}$, $\mathbb{A}^{(n)}$, etc.), and those that serve mostly as elements of other sets (such as particular atoms, tuples of atoms, etc.). The former will usually be equivariant, and will be denoted by capital letters X, Y, A, and referred to as *sets with atoms*. The latter will often have nonempty least support, and will be denoted by small letters x, y, a, b, etc., and referred to as *elements*.

Orbit-finite sets. Elements naturally fall into disjoint *orbits*: x and y are in the same orbit if $\pi(x) = y$ for some bijection $\pi : \mathbb{A} \to \mathbb{A}$. A set is equivariant if and only if it is a union of orbits. For example, $\mathbb{A}$, $\mathbb{A}^{(n)}$ and $\binom{\mathbb{A}}{n}$ comprise one orbit each. The set $\mathbb{A}^2$ decomposes into two orbits – the diagonal and its complement – and $\mathbb{A}^*$ and $\mathcal{P}_{\text{fin}}\mathbb{A}$ have infinitely many orbits. In sets with atoms, sets with finitely many orbits (or *orbit-finite sets*) play the role of finite sets. The *dimension* of a set A, denoted $\dim A$, is the maximal size of the support of any of its elements. Every orbit-finite set has a finite dimension.

For every element x there is a unique single-orbit set – called *the orbit* of x – which contains x, namely the set of all elements of the form $\pi(x)$, where π is an atom permutation. For any finite set of elements there is a smallest equivariant set containing it, which is orbit-finite. Every orbit-finite set arises in this way, so orbit-finite sets indeed are "finite up to atom renaming".

Automorphisms of elements. Let x be an element. A permutation π of $\sup x$ is an *automorphism* of x if x is fixed by some (equivalently, every) permutation of atoms which extends π. Automorphisms of x form a group of permutations of $\sup(x)$, denoted $\mathrm{Aut}(x)$. For example, if x is an unordered pair $\{a, b\}$ of distinct atoms, then $\mathrm{Aut}(x)$ is the symmetric group on $\{a, b\}$. On the other hand, if x is the *ordered* pair (a, b), then $\mathrm{Aut}(x)$ is the trivial group acting on $\{a, b\}$. If x is a finite relational structure with atoms as nodes, such as a graph, then $\mathrm{Aut}(x)$ is the classical automorphism group of x.

When x and y are in the same orbit then $\mathrm{Aut}(x)$ and $\mathrm{Aut}(y)$ are isomorphic as permutation groups. The converse almost holds: if $\mathrm{Aut}(x)$ and $\mathrm{Aut}(y)$ are isomorphic as permutation groups, then the orbit of x maps equivariantly and bijectively to the orbit of y. This means that orbit-finite sets can be presented (up to equivariant bijection) by finite collections of finite permutation groups. Moreover, properties of orbit-finite sets, such as the standardness of orbit-finite alphabets, can be considered as properties of finite permutation groups.

Turing machines. Following [4], a Turing machine with atoms (TMA) is defined exactly as an ordinary Turing machine, but with finite sets replaced by orbit-finite sets with atoms. Thus a TMA consists of an input alphabet A, a work alphabet $B \supseteq A$, and set of states Q with distinguished subsets of initial and accepting states, all these orbit-finite sets with atoms, and an equivariant transition relation

$$\delta \subseteq Q \times B \times Q \times B \times \{-1, 0, 1\}$$

where the last atomless component encodes possible moves of the machine head as usual. An input is a finite word $w \in A^*$, and the definitions of a machine configuration, transition between configurations, machine run, acceptance and the language recognized by a machine are as in the classical case. A machine is deterministic if the transition relation is a partial function and there is exactly one initial state.

Some examples of TMAs were given in [4]; we follow with a few more to illustrate TMA determinization issues.

Example 2.1. Consider the alphabet $\mathbb{A}^2$ of ordered pairs of atoms; a word over this alphabet may be seen as a finite directed (multi-)graph with atoms as vertices. TMAs can decide all standard graph-theoretic properties of such words in a uniform way. Indeed, let the working alphabet of a machine M additionally contain single atoms as letters. The machine, given an input word w, may begin by deterministically computing (and writing to its tape) an ordered list of all atoms in w, in the order of appearance. This is done by checking, for each letter (a, b) of the input, whether a (and, further, b) appear in the list constructed so far, and if not, by adding them to the list. For this, it is important that a deterministic TMA, given a letter (a, b), may compute the atoms a and b; indeed, both projections from $\mathbb{A}^2$ to $\mathbb{A}$ are equivariant functions.

Once an ordered list of all atoms in w is computed, the machine M may simulate any classical, atomless algorithm on finite directed (multi-)graphs, representing every atom in w by the number of its position in the list. If the simulated algorithm is deterministic, then so is M.

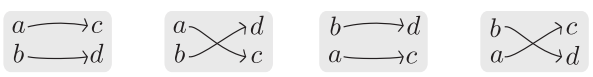
Example 2.2. Consider now the alphabet $\binom{\mathbb{A}}{2}$ of *unordered* pairs of atoms. By analogy to the previous example, a word may now be seen as a finite *undirected* (multi-)graph. The simple approach sketched above does not work as it is. Indeed, as we explained before, there is no equivariant function that, given a letter $\{a, b\}$, returns the atom a. As a result, a deterministic TMA cannot, in general, compute a total order of atoms that appear in a word over $\binom{\mathbb{A}}{2}$.

Fortunately, in this case the problem may be overcome rather easily. Note that taking the intersection, or the difference, of two sets of atoms is an equivariant function. Therefore, if some atom appears in one letter but not in another, then a deterministic TMA can detect this, and output this atom at the end of the tape. This way, the machine outputs all vertices of all non-isolated edges, in some order. Based on this order, a machine may again simulate any classical algorithm on finite undirected (multi-)graphs as in Example 2.1, with the only difference that it must remember that the input graph has a certain number of isolated edges that are not represented in the computed list of atoms.

Example 2.3. Consider the alphabet A of unordered pairs of disjoint, ordered pairs of atoms. Its letters are of the form

$$l = \{(a, c), (b, d)\}$$

for distinct $a, b, c, d \in \mathbb{A}$, and we may draw them as graphs:



Above, the same letter is depicted in four different ways, depending on the ordering of atoms chosen in the picture. It has a four-element support and two automorphisms:

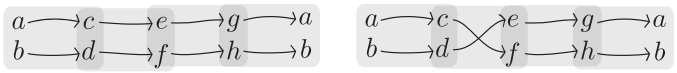
$$\sup(l) = \{a, b, c, d\} \qquad \mathrm{Aut}(l) = \{(),\ (a\ b)(c\ d)\}.$$

Here and in the following, we use the usual cycle decomposition notation to describe finite permutations.

Given a letter as above, we shall call the set $\{a, b\}$ its *left bag* and $\{c, d\}$ its *right bag*. Consider the language $L \subseteq A^*$ of words $l_1 l_2 \cdots l_n$ such that:

(i) for $1 \leq i < n$, the right bag of l_i equals the left bag of l_{i+1},
(ii) the right bag of l_n equals the left bag of l_1,
(iii) otherwise, bags are pairwise disjoint, and
(iv) it is possible to choose one pair of atoms (an "edge") from each l_i so that the chosen edges form a directed cycle of length n.

For example, the four-letter word on the left belongs to L, and the one on the right does not, as it fails the condition (iv):



An equivalent (slightly informal) phrasing of condition (iv) is that the atoms can be arranged in such a way, that the letters have no crossings.

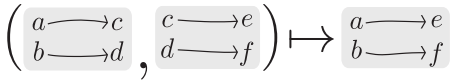
To investigate the recognizability of L, note that functions that return the left and the right bag of a given letter:

$$\{a, b\} \hookleftarrow \{(a, c), (b, d)\} \mapsto \{c, d\}$$

are equivariant, and therefore computable in a single step by a deterministic TMA. Since comparing two bags for equality (and checking whether they are disjoint) is also deterministically computable, it is clear that a deteministic TMA can check conditions (i)-(iii) in the definition of L. These conditions ensure that the input word has the shape of a circular band; the remaining condition (iv) says that it is a simple band, and not a "Möbius strip".

A *non*deterministic TMA can check the condition (iv) easily, guessing one edge from each letter of the input, writing them on the tape, and then deterministically checking that they form a directed cycle. For this, the work alphabet should be extended to include single edges, i.e., ordered pairs of atoms.

Although a deterministic TMA is unable to guess an edge from a letter of the alphabet A, condition (iv) can still be checked deterministically. To this end, note that two letters of A that share a bag, may be deterministically composed with an equivariant function that acts as follows:



A deterministic machine can sequentially compose the input letters from the input in this way, storing intermediate results in its state, and finally check that the structure obtained at the end is of the form:

.

The above deterministic construction relies on the fact that non-local dependencies on bags in the input word may be encoded as small structures of atoms (i.e. intermediate results of the composition process). In [4], an alphabet was proposed where this fortunate property fails, and as a consequence, TMAs do not determinize. We briefly recall that example now.

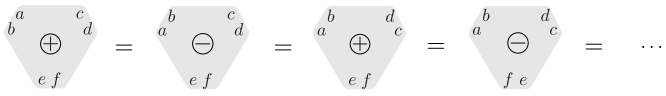
Example 2.4. Generalizing Example 2.3, consider an alphabet A whose letters are four-element sets of atom triples of the following shape, containing six atoms altogether:

$$l = \{(a, c, e), (a, d, f), (b, c, f), (b, d, e)\}.$$

It is straightforward to check that l has four automorphisms:

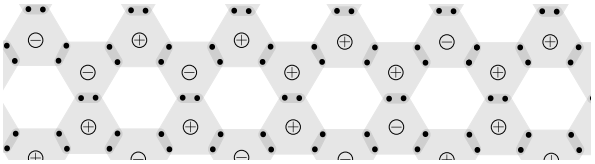
$$\sup(l) = \{a, b, c, d, e, f\}$$
$$\mathrm{Aut}(l) = \{(),\ (a\ b)(c\ d),\ (a\ b)(e\ f),\ (c\ d)(e\ f)\}.$$

A letter like this may be depicted in eight different ways, as a hyperedge on six vertices, with a positive or negative sign:



The rule is that each time a pair of atoms at some corner exchanges positions, the sign changes. This is analogous to Example 2.3, where each time a pair exchanges positions, then a crossing in the graph either appears or disappears.

In the alphabet of Example 2.3, an automorphism of a letter could swap a pair of atoms in a bag if and only if it swapped the remaining atoms as well. Here, the situation is a little more complicated: there are three exchangeable pairs of atoms in a letter, and an automorphism can perform a swap in any two (but not all three) of them. This enables much more complicated dependencies between bags. In [4], letters of A were used to cover a surface (specifically, a torus) as depicted below (atoms are represented by dots):

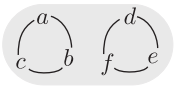


One then considers the language of words whose letters form such a torus (where the correct shape is ensured by conditions analogous to (i)-(iii) in Example 2.3) and satisfy an additional requirement analogous to (iv) in Example 2.3: that one can swap the atoms in such a way that each letter gets a positive sign. Note that swapping a pair of atoms changes the signs of both letters which contain this pair. A nondeterministic TMA can recognize this language similarly to Example 2.3, but, as is proved with a geometric argument, a deterministic TMA cannot recognize it. More details on this can be found in [4].

Example 2.5. Another way of arranging six atoms in a letter is a two-element set of disjoint three-element sets:

$$l = \big\{\{a, b, c\}, \{d, e, f\}\big\}$$

which may be presented as an undirected graph:



A letter like this has support $\{a, b, c, d, e, f\}$ and $3! \cdot 3! \cdot 2 = 72$ automorphisms. As we will show in Section 5, TMAs over this alphabet A do not determinize. On the other hand, as will also easily follow from our results, machines over *ordered pairs* of disjoint three-element sets of atoms, or over two-element sets of disjoint *ordered triples* of atoms, do determinize.

An alphabet will be called *standard* if all nondeterministically recognizable languages over it are also deterministically decidable; otherwise it is *nonstandard*. Our aim is to provide an effective characterization of these properties.

3. Word isomorphism and (k, l)-consistency

Recall that for any element x, an *automorphism* of x is a bijection on its support which extends to a bijection of atoms that fixes x. More generally, an *isomorphism* from x to y is a bijection from $\sup(x)$ to $\sup(y)$ which extends to an atom bijection that maps x to y. Note that x and y are isomorphic in this sense if and only if they are in the same orbit.

The *word isomorphism problem* for an orbit-finite alphabet A is the language

$$\mathit{Iso}_A = \{vw \in A^* \mid v \text{ is isomorphic to } w\}.$$

We recall the following theorem from [4]:

Theorem 3.1. *An alphabet A is standard if and only if the language Iso_A is decidable by a deterministic TMA.*

We will now study the word isomorphism problem in more detail. As a first step to test isomorphism, a TMA may determine whether two words are *similar*, as described below.

3.1 Word similarity

Bags. We say that two atoms *coappear* in a word $w \in A^*$ if they belong to its support and belong to the supports of exactly the same letters of w. Coappearance is an equivalence relation on $\sup(w)$, and its equivalence classes will be called *bags* of w, a notion that appeared in Example 2.3. A bag is either disjoint from or contained in the support of any letter of w, moreover it is determined by the sequence of letters which contain it. The set of all bags in w will be denoted by $\mathrm{Bags}(w)$. It inherits a total ordering from the lexicographic ordering on subsequences of w.

Similar words. Suppose that two words v and w over A have the same length and the same number of bags. We say that a letter of v *corresponds* to a letter of w if they are on the same positions in those words. Similarly a bag B in v *corresponds* to a bag $\tilde{B}$ in w if for every letter of v which contains B, the corresponding letter in w contains $\tilde{B}$, and vice versa.

We say that the words v and w are *similar* if each bag of v has a corresponding bag of the same size in w, and vice versa. Similar words induce a bijective correspondence between their bags. When v and w are understood from the context, we denote by $\tilde{B}$ the bag in w corresponding to a bag B in v.

Lemma 3.2. *There exists a deterministic TMA which determines whether two input words $w, v \in A^*$ are similar.*

Note that similar words are not necessarily isomorphic, as witnessed by the two four-letter words in Example 2.3.

3.2 Consistency

We now go beyond local interactions between intersecting letters and proceed to a more refined analysis of the structure of the words v and w. In this section, fix two similar words v and w over A.

Translations. An isomorphism σ from v to w decomposes into a family of bijections – one bijection σ_B from B to $\tilde{B}$ per each bag in v. In this section, we study families of bijections between corresponding bags which are "candidates" for forming a global isomorphism from v to w.

Let $\mathcal{B} = \{B_1, \ldots, B_n\}$ be some family of bags of v. A *translation* on *domain* $\mathcal{B}$ is a family of bijections $\sigma = (\sigma_B)_{B \in \mathcal{B}}$, where σ_B is a bijection from B to $\tilde{B}$. The *size* of σ is the size of its domain. A translation σ *covers* a letter l of the word v, if its domain contains all the bags of l.

Local consistency. A translation σ is *locally consistent* if it induces an isomorphism of every letter of v which it covers to the corresponding letter in w. Note that an isomorphism between v and w induces a locally consistent translation on the domain of all bags of v. Conversely, any locally consistent translation on that domain gives an isomorphism from v to w.

Example 3.1. Consider the pair of words v, w from Example 2.3. Both have four bags: $\{a, b\}, \{c, d\}, \{e, f\}, \{g, h\}$. For $i = 1, 2, 3, 4$, let σ_i map the ith bag of v identically to the ith bag of w. The translation (σ_1, σ_2) is locally consistent: it covers the first letter of v, and maps it isomorphically to the corresponding letter of w. However, the translation $(\sigma_1, \sigma_2, \sigma_3)$ is not locally consistent, as it fails to map the second letter of v to the second letter of w.

A consistency algorithm. We now sketch a variant of the (k, l)-consistency algorithm (see e.g. [2, 8]), adjusted to finding word isomorphisms. The relationship to the standard (k, l)-consistency algorithm will become clear in Sec. 4.

The algorithm has two natural numbers $k < l$ as parameters, and takes a pair of similar words v and w as input.

Let $\mathcal{F}$ be a collection of locally consistent translations of size at most l. We say that $\mathcal{F}$ is (k, l)-*consistent* if:

(1) $\mathcal{F}$ is *downward-closed*: if $\sigma \in \mathcal{F}$ is a translation and $\mathcal{B}$ is a subset of its domain, then the restriction $(\sigma_B)_{B \in \mathcal{B}}$ is in $\mathcal{F}$.

(2) $\mathcal{F}$ is *weakly upward-closed*: if $\sigma \in \mathcal{F}$ is a translation of size at most k and $\mathcal{B}$ is a family of at most l bags of v that contains the domain of σ, then there is a translation $\bar{\sigma} \in \mathcal{F}$ with domain $\mathcal{B}$ which extends σ.

Given two input words v, w, the (k, l)-*consistency algorithm* computes the largest (k, l)-consistent collection of locally consistent translations. The algorithm starts with the collection of all locally consistent translations of size at most l, and repeatedly removes all translations σ that falsify condition (1) or (2), until a fixpoint is reached. The result of this procedure is denoted $\mathit{cons}_{k,l}(vw)$.

The algorithm for computing $\mathit{cons}_{k,l}(vw)$ can be carried out by a deterministic TMA, where letters of the work alphabet include families of consistent translations over a common domain of size at most l. This work alphabet is (contained in) an orbit-finite set, and admits the operations needed for the fixpoint computation of $\mathit{cons}_{k,l}(vw)$.

If v and w are isomorphic then $\mathit{cons}_{k,l}(vw) \neq \emptyset$. Indeed, an isomorphism from v to w induces a locally consistent translation on the domain of all bags, and all its subfamilies of size at most l form a (k, l)-consistent collection.

We say that the alphabet A has *width* (k, l) if the other implication holds, i.e., if for any pair of similar words $v, w \in A^*$, $\mathit{cons}_{k,l}(vw) \neq \emptyset$ if and only if v and w are isomorphic.

The following theorem says that the (k, l)-consistency algorithms are in a sense "universal" for recognizing Iso_A.

Theorem 3.3. *For any alphabet A, the language Iso_A is recognized by a deterministic TMA if and only if there exist numbers k, l such that A has width (k, l).*

One implication is easy: if A has width (k, l) then the language Iso_A is deterministically recognizable (in polynomial time) by the TMA which computes $\mathit{cons}_{k,l}(vw)$ and tests whether the result is nonempty. For the other implication, suppose that Iso_A is recognized by a deterministic TMA M and that the family $\mathcal{F} = \mathit{cons}_{k,l}(vw)$ is nonempty (for sufficiently large k, l, depending on M). Roughly, one then shows that an accepting run of M over the word vv can be translated, using the family $\mathcal{F}$, into an accepting run over vw, implying that $vw \in \mathit{Iso}_A$. Details can be found in Appendix A.

Together with Theorem 3.1, this gives a characterization of standard alphabets in terms of width. However, it may not be clear how the existence of a finite width might be decided. In the next section, we encode the existence of an isomorphism between similar words as a constraint satisfaction problem, to draw on the rich body of results known about those.

4. Constraint Satisfaction Problems

An *instance* of a Constraint Satisfaction Problem (CSP) [8] consists of a set of *variables*, a set of *values*, called its *domain*, and a family of *constraints*. Each constraint is of the form $(\bar{v}, R)$, where $\bar{v} = (v_1, \ldots, v_n)$ is a tuple of variables, and R is an n-ary relation over the domain; we say that the tuple $\bar{v}$ is *constrained* to R. For a given instance, a *partial assignment* is a partial mapping of the variables to the domain. A partial assignment f is called a *partial solution* if it satisfies all the constraints, i.e., if $\bar{v}$ is constrained to R and f is defined over $\bar{v}$, then $(f(v_1), \ldots, f(v_n)) \in R$. We drop the qualifier *partial* if the mapping is total.

We will be interested in the case where all the above sets are finite.

4.1 Instances

Fix an orbit-finite alphabet A. The aim is to reduce the word isomorphism problem over A to a CSP. More precisely, given two similar words v and w over A, we will construct an instance which has a solution if and only if v and w are isomorphic. The idea is that each bag in v is a variable, and an assignment will assign to it a bijection to the corresponding bag in w. In order to describe bijections between bags using elements of a finite domain, we need to introduce canonical representations of various bags. The precise definition follows.

Maps and atlases. For a bag B, define $[B] = [n] = \{1, 2, \ldots, n\}$ where n is the size of B. A *map* of a bag B is any bijection from B to $[B]$, or equivalently, an ordering of the elements of B. An *atlas* α on a word v over A is a family of maps: one map α_B per each bag B of v.

Fix two similar words v and w over A, and let α and β be their atlases. Then $[B] = \left[\tilde{B}\right]$ for any bag B in v, and any permutation τ_B of $[B]$ induces a bijection σ_B from B to $\tilde{B}$:

$$\sigma_B = \alpha_B \cdot \tau_B \cdot (\beta_{\tilde{B}})^{-1} \tag{1}$$

(we use left-to-right function composition here). The correspondence of σ_B and τ_B is bijective. Our aim is to define a CSP instance whose solutions are families $(\tau_B)_{B \in \mathrm{Bags}(v)}$, where each τ_B is a permutation of $[B]$, such that the corresponding translation $(\sigma_B)_{B \in \mathrm{Bags}(v)}$ is locally consistent, and so induces an isomorphism from v to w.

The instance. For fixed atlases α and β, we define the following instance, denoted $I^{v,\alpha}_{w,\beta}$. Its variables are all bags of v. Its domain is the disjoint union:

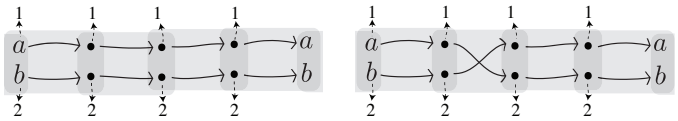
$$D_A = \mathcal{S}_{[1]} + \mathcal{S}_{[2]} + \mathcal{S}_{[3]} + \cdots + \mathcal{S}_{[\dim A]} \tag{2}$$

where $\mathcal{S}_{[n]}$ is the group of all permutations of $[n]$. Note that $\dim A$ is the maximal possible size of a bag in a word over A.

Most importantly, there are the constraints. They correspond to letters of v as follows. For a letter l of v, let $B_1, \ldots, B_n$ be its bags in increasing order. The tuple $(B_1, \ldots, B_n)$ is constrained to the set R_l of tuples $(\tau_1, \ldots, \tau_n) \in \mathcal{S}_{[B_1]} \times \cdots \times \mathcal{S}_{[B_n]}$ such that the translation $(\sigma_1, \ldots, \sigma_n)$ induced via (1) is locally consistent, i.e., forms an isomorphism from l to the corresponding letter of w. Note that R_l can be seen as an n-ary relation over D_A. Such are the constraints of the instance $I^{v,\alpha}_{w,\beta}$.

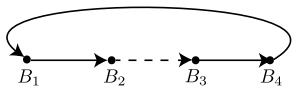
The construction implies that partial assignments for $I^{v,\alpha}_{w,\beta}$ correspond bijectively to translations from v to w. Moreover, a partial assignment is a partial solution if and only if the corresponding translation is locally consistent. The correspondence preserves inclusions of partial assignments and translations.

Example 4.1. Over the alphabet consisting of letters $\{(a,c),(b,d)\}$, consider the two words v, w from Example 2.3, depicted below with most atoms represented by dots. Some atlases α, β of v, w are given; they assign numbers to atoms.



The instance has four variables B_1, B_2, B_3, B_4 corresponding to the four bags of v. There are four constraints, corresponding to the letters of v. The first letter constrains the pair (B_1, B_2) to the set of pairs $R = \{(id, id), (\sigma, \sigma)\} \subseteq \mathcal{S}_{[2]} \times \mathcal{S}_{[2]}$ (where σ denotes the transposition on $[2]$) because only these correspond to isomorphisms from the first letter of v to the first letter of w. The pair (B_2, B_3), however, is constrained to the set of pairs $R' = \{(id, \sigma), (\sigma, id)\}$.

The resulting instance $I^{v,\alpha}_{w,\beta}$ is drawn below. Solid edges represent the constraint R, the dashed edge – the constraint R'.



For different atlases on the same v and w the instance may look different, but it will always have one or three dashed edges.

4.2 Templates

If $T = (D, R_1, R_2, \ldots, R_n)$ is a relational structure, then we say that an instance is *over the template* T if its domain is D and each relation occurring in a constraint is one of the relations $R_1, \ldots, R_n$.

Fix an orbit-finite alphabet A. We define the template T_A as the set D_A from (2) together with all relations on D_A which appear in the constraints of all instances of the form $I^{v,\alpha}_{w,\beta}$. There are only finitely many such relations because each of them has arity at most $\dim A$. Moreover, each relation in T_A has a special structure: it is a coset.

If G is a group, then a *coset* in G is any subset of the form $H \cdot g = \{h \cdot g : h \in H\}$, for a subgroup H of G and $g \in G$.

Lemma 4.1. *Let v, w be similar words over A and α, β their atlases. For any letter l of v with bags $B_1, \ldots, B_n$, the relation R_l is a coset in the group $\mathcal{S}_{[B_1]} \times \cdots \times \mathcal{S}_{[B_n]}$, and a subgroup if $v = w$ and $\alpha = \beta$.*

Example 4.2. Let A be the alphabet from the previous example. Then $D_A = \mathcal{S}_{[1]} + \mathcal{S}_{[2]} + \mathcal{S}_{[3]} + \mathcal{S}_{[4]}$. Some of the relations of T_A include the binary relations $R, R' \subseteq \mathcal{S}_{[2]} \times \mathcal{S}_{[2]}$ which appeared in Example 4.1. Note that R is a subgroup and R' is its coset in $\mathcal{S}_{[2]} \times \mathcal{S}_{[2]}$. Other relations in T_A include a unary relation U which is a subgroup of $\mathcal{S}_{[4]}$ isomorphic to $\mathrm{Aut}(l)$, where l is a letter of A. The relation U arises in the instance $I^{l,\alpha}_{l,\alpha}$, where α is some map on l.

4.3 Templates of bounded width

Given an instance I, the well-known (k,l)-consistency algorithm (see e.g. [2, 8]) is completely analogous to the one in Sec. 3.2, but it computes partial solutions instead of locally consistent translations. We say that a template T has *width* (k,l) if for any instance I over T, $\mathit{cons}_{k,l}(I) \neq \emptyset$ if and only if I has a solution. A template of *bounded width* is a template of width (k,l), for some natural numbers k, l.

By construction of the instance $I^{v,\alpha}_{w,\beta}$, its partial solutions correspond to locally consistent translations, so we have:

Fact 4.2. *For any words v, w over A and their atlases α, β, $\mathit{cons}_{k,l}(vw) \neq \emptyset$ if and only if $\mathit{cons}_{k,l}(I^{v,\alpha}_{w,\beta}) \neq \emptyset$.*

It is also not difficult to prove the following.

Proposition 4.3. *Fix an alphabet A, and let k, l be natural numbers. Then the following conditions are equivalent:*

1. *The alphabet A has width (k,l),*
2. *The template T_A has width (k,l).*

The implication from (2) to (1) is immediate from Fact 4.2, since any pair of words v, w induces an instance $I^{v,\alpha}_{w,\beta}$ over T_A, and the consistency algorithm over $I^{v,\alpha}_{w,\beta}$ simulates the algorithm over vw. The converse implication is similar, and uses the the fact that an arbitrary instance I over T_A can be converted into a homomorphically equivalent instance of the form $I^{v,\alpha}_{w,\beta}$. The details are in Appendix B.

4.4 Majority polymorphisms

The bounded width property is decidable for any template [2, 13], but for templates T_A it can be analyzed further using the following fundamental notions of CSP theory.

Consider a template T over a domain D and a function $f : D^n \to D$. A relation R in T is *compatible with* f if by applying f (coordinatewise) to n tuples from R, we get a tuple in R. We say that f is a *polymorphism* of T if all the relations in T are compatible with f. A *majority* polymorphism m is a ternary polymorphism such that for all $x, y \in D$ we have

$$m(x,x,y) = m(x,y,x) = m(y,x,x) = x. \tag{3}$$

Another example of a ternary polymorphism is a *Maltsev* polymorphism M which for all $x, y \in D$ satisfies

$$M(x,x,y) = M(y,x,x) = y. \tag{4}$$

For every alphabet A, the template T_A has a Maltsev polymorphism M defined by:

$$M(x,y,z) = xy^{-1}z, \quad \text{if } x,y,z \in \mathcal{S}_{[n]} \text{ for some } n,$$

and for other arguments defined arbitrarily but satisfying (4). This is a polymorphism since, by Lemma 4.1, every relation in T_A is a coset.

A relational structure is a *core* if every endomorphism of it is a bijection. Every relational structure T has an induced substructure C which is a core and which is a retract of T, i.e. there is a homomorphism from T onto C which is the identity on C. We call C *the core* of T (a core is unique up to isomorphism).

The key technical result of CSP theory that we need is:

Theorem 4.4 ([7])**.** *If a core template has a Maltsev polymorphism and has bounded width, then it has a majority polymorphism.*

The following lemma does not hold for arbitrary relational templates, but it does for templates T_A:

Lemma 4.5. *For any alphabet A, the template T_A has a majority polymorphism if and only if its core has a majority polymorphism.*

From this we deduce:

Lemma 4.6. *For any alphabet A, the template T_A has bounded width if and only if T_A has a majority polymorphism.*

Proof. The right-to-left implication is classical [8] and holds for any template T. To prove the left-to-right implication, let C be the core of T_A, and let r be a homomorphism of T_A onto C which is the identity on C. Then C has a Maltsev polymorphism $r \circ M$, where M is a Maltsev polymorphism of T_A. It is well known (and easy to see) that a template has bounded width if and only if its core has; therefore, C has bounded width. By Theorem 4.4, C

has a majority polymorphism. Therefore, by Lemma 4.5, T_A has a majority polymorphism. □

We arrive at the main result of this paper:

Theorem 4.7. *An alphabet A is standard if and only if its template T_A has a majority polymorphism.*

Proof. Follows from Theorems 3.1 and 3.3, Proposition 4.3, and Lemma 4.6. □

5. Applications

Theorem 4.7 gives a decidable (in the classical sense) characterization of standard alphabets. Indeed, given A represented as a finite collection of finite permutation group, the template T_A can be computed, since all relations in it arise from words of bounded size. Then one can try every ternary function on the domain of T_A for the polymorphism property and majority equations. (Note that there are also more efficient, polynomial time procedures of testing whether a template has bounded width [2, 13].) Rather than illustrate this tedious procedure, we show on examples how the existence of a majority polymorphism, or the lack thereof, may be deduced by studying the structure of a template T_A.

5.1 Alphabets of dimension up to five are standard

We will show that any alphabet A of dimension (at most) five is standard. The domain of the template T_A of such an alphabet has (at most) five components:

$$D_A = \mathcal{S}_{[1]} + \mathcal{S}_{[2]} + \mathcal{S}_{[3]} + \mathcal{S}_{[4]} + \mathcal{S}_{[5]}.$$

By definition of T_A, every relation in it is of the form:

$$R \subseteq \mathcal{S}_{[n_1]} \times \mathcal{S}_{[n_2]} \times \cdots \times \mathcal{S}_{[n_k]} \tag{5}$$

where $\sum_{i=1}^{k} n_i \leq 5$. Moreover, R is always a coset in the product group on the right.

For a function $m : D_A^3 \to D_A$ to be a polymorphism, it is required that every relation R in T_A is compatible with m, i.e., that whenever the first three rows below are tuples in R, the resulting tuple also is in R, where $\rho_i = m(\sigma_i, \tau_i, \nu_i)$:

$$m \begin{array}{|cccccc} \sigma_1 & \sigma_2 & \sigma_3 & \cdots & \sigma_k & \in R \\ \tau_1 & \tau_2 & \tau_3 & \cdots & \tau_k & \in R \\ \nu_1 & \nu_2 & \nu_3 & \cdots & \nu_k & \in R \\ \hline \rho_1 & \rho_2 & \rho_3 & \cdots & \rho_k & \in R. \end{array}$$

We will show that there exists a majority function for which *all* coset relations R as in (5) are compatible, thereby proving that every alphabet of dimension A is standard.

According to (5), every relation R determines a tuple of components $S_{[n_i]}$ so that ith element of a tuple in R belongs to the ith component. As a result, to define a polymorphism m on T_A, it is enough to define $m(x, y, z)$ for x, y and z in the same $\mathcal{S}_{[n]}$. On the remaining triples m may be defined arbitrarily, without compromising the polymorphism property.

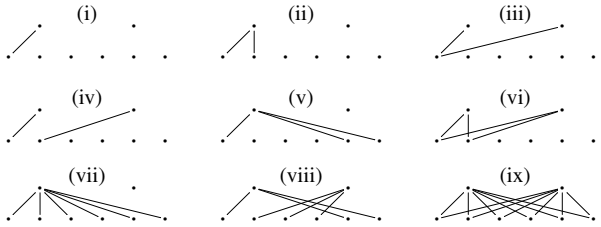
We shall now study the structure of coset relations R as in (5) in some more detail. First, consider a relation R where some n_i in (5) equals 1, and let R_{-i} be the projection of R to all coordinates except i. Clearly, R_{-i} is a coset relation. It is easy to see that R is compatible with a given majority function m if and only if R_{-i} is. As a result, such relations R may safely be excluded when checking the polymorphism property on all coset relations.

On the other hand, assume R is unary. Call a ternary function m *conservative* if $m(\sigma, \tau, \nu) \in \{\sigma, \tau, \nu\}$ for all arguments σ, τ and ν. It is easy to see that all unary relations are compatible with every conservative function. Therefore, for any conservative m, we may disregard all unary relations in T_A as well.

Now consider binary relations $R \subseteq \mathcal{S}_{[2]} \times \mathcal{S}_{[2]}$ in T_A. $\mathcal{S}_{[2]}$ has two elements, so there is only one way to define a majority function on this component: the majority equations determine it fully. It is easy to check that every binary relation on a two-element set is compatible with the unique majority function on it, so such relations may also be disregarded.

The only remaining case is binary relations $R \subseteq \mathcal{S}_{[2]} \times \mathcal{S}_{[3]}$. Arranging the elements of $\mathcal{S}_{[2]}$ and $\mathcal{S}_{[3]}$ in a diagram:

we may draw any R as a bipartite graph. Recall that any R in T_A is a subgroup or a coset in the group $\mathcal{S}_{[2]} \times \mathcal{S}_{[3]}$. Let us consider subgroups first; all are depicted here (dots correspond to permutations as drawn above)



with the exception of two conjugate subgroups per each of (ii), (iv) and (vi), which are isomorphic to them as bipartite graphs. Recall that there is only one way to define a majority function on $\mathcal{S}_{[2]}$; the question is whether we can extend it with a majority function m on $\mathcal{S}_{[3]}$ so that all relations above are compatible with it. It is easy to see that, as far as such exentions are concerned:

- (i)-(iv) and (vi) are compatible with any majority function,
- (v) is compatible with any conservative function,
- (vii) and (ix) are compatible with any function.

The only remaining case is (viii). The group $\mathcal{S}_{[3]}$ contains three even permutations $()$, $(1\,2\,3)$ and $(1\,3\,2)$, and three odd ones $(1\,2)$, $(1\,3)$, and $(2\,3)$. We say that a function $m : \mathcal{S}_{[3]}^3 \to \mathcal{S}_{[3]}$ is *odd-even-majority* if $m(\pi_1, \pi_2, \pi_3)$ is even whenever at least two of π_1, π_2, π_3 are even, and odd otherwise. It is easy to check that:

- (viii) is compatible with any odd-even-majority function.

Cosets in $\mathcal{S}_{[2]} \times \mathcal{S}_{[3]}$ do not pose additional problems. Indeed, all cosets of (i)-(vii) are isomorphic to their respective subgroups as bipartite graphs, and they are compatible with functions listed above for their respective subgroups. The full group (ix) has no cosets, and the unique coset of (viii) is its complement as a bipartite graph, and it is compatible with every odd-even-majority function.

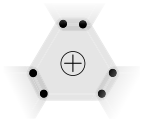
Altogether we have proved that every ternary function on D_A that is conservative, a majority, and an odd-even-majority when restricted to $\mathcal{S}_{[3]}^3$, is a polymorphism on T_A. It is easy to define a function with all these properties, and as a result, T_A has a majority polymorphism, so A is standard.

5.2 Some nonstandard alphabets of dimension six

As one could expect, there are standard alphabets of dimensions greater than five. For instance, the alphabet $\binom{\mathbb{A}}{k}$ of k-element sets of atoms is standard for any k. It is easy to see this by looking at their templates, but there is an even simpler argument: over such an alphabet, two words are isomorphic if and only if they are similar.

However, there are nonstandard alphabets of dimension six. Consider the alphabet from Example 2.4. We depict a fragment of a word v over this alphabet (with atoms anonymized as dots as in

Example 4.1), where the letter in the center intersects three other letters, splitting it into three bags:



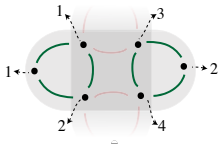
Consider the instance $I^{v,\alpha}_{v,\alpha}$, where α is any atlas on v (its choice will not affect the following). The triple of bags is constrained to a ternary relation R on $\mathcal{S}_{[2]}$ that consists of those triples of permutations where the swap $\sigma = (1\ 2) \in \mathcal{S}_{[2]}$ appears zero or two times.

As $\mathcal{S}_{[2]}$ has only two elements, there is exactly one majority function m on $\mathcal{S}^3_{[2]}$. As it turns out, it is not compatible with R:

$$m\begin{array}{cccc} () & \sigma & \sigma & \in R \\ \sigma & () & \sigma & \in R \\ \sigma & \sigma & () & \in R \\ \hline \sigma & \sigma & \sigma & \notin R, \end{array}$$

hence m is not a polymorphim of T_A and the alphabet from Example 2.4 is nonstandard. The reader should enjoy comparing this short argument with the pain suffered in [4] to prove the same result directly, without using CSP theory.

Now consider Example 2.5, and a fragment of a word v with a letter split into two bags of size two and four (the left and right atoms are in one bag), together with an atlas α as below:



In the instance $I^{v,\alpha}_{v,\alpha}$, the two bags of this letter are constrained to a binary relation $R_{\{1,2\}} \subseteq \mathcal{S}_{[4]} \times \mathcal{S}_{[2]}$ which is the graph of the partial function $f_{\{1,2\}} : \mathcal{S}_{[4]} \to \mathcal{S}_{[2]}$ such that

$$f_{\{1,2\}}(\pi) = \begin{cases} () & \text{iff } \pi \text{ preserves the set } \{1,2\}, \\ \sigma & \text{iff } \pi \text{ maps } \{1,2\} \text{ to } \{3,4\}. \end{cases}$$

For the same word but different atlases other relations $R_{\{1,3\}}$ and $R_{\{1,4\}}$, which are graphs of partial functions $f_{\{1,3\}}$ and $f_{\{1,4\}}$ defined analogously to $f_{\{1,2\}}$, can be obtained.

Assume any majority function m on the domain of T_A. For the following permutations in $\mathcal{S}_{[4]}$:

$$\pi_1 = (1\ 2)(3\ 4) \qquad \pi_2 = (1\ 3)(2\ 4) \qquad \pi_3 = (1\ 4)(2\ 3)$$

consider the value $\chi = m(\pi_1, \pi_2, \pi_3)$. Since $f_{\{1,2\}}(\pi_1) = ()$, $f_{\{1,2\}}(\pi_2) = \sigma$, and $f_{\{1,2\}}(\pi_3) = \sigma$, if m where a polymorphism then we would have $f_{\{1,2\}}(\chi) = m((), \sigma, \sigma) = \sigma$. Analogously we can prove that $f_{\{1,3\}}(\chi) = f_{\{1,4\}}(\chi) = \sigma$. This means that the permutation χ maps each of sets $\{1,2\}, \{1,3\}, \{1,4\}$ to their complements, which is impossible. As a result, m cannot be a polymorphism, hence the alphabet of Example 2.5 is nonstandard.

6. Descriptive Complexity

We relate our results concerning TMAs to logics over relational structures. To this end, we first describe a class of finite relational structures which correspond to words with atoms. We then describe how TMAs over such words correspond, in terms of expressive power, to certain logics over these relational structures.

In this section we consider both classical Turing machines and TMAs, but only deterministic ones.

6.1 Linearly patched structures

Below we consider finite relational structures over a finite vocabulary, assuming without loss of generality that the domains of the structures are subsets of the set $\mathbb{A}$ of atoms.

A *patched* structure is a relational structure $\mathfrak{M}$ together with a collection of substructures of $\mathfrak{M}$ (not necessarily induced substructures) which we call *patches* of $\mathfrak{M}$, such that $\mathfrak{M}$ is covered by its patches, i.e. the domain and relations of $\mathfrak{M}$ are the set-theoretical unions (not necessarily disjoint) of the domains and relations of its patches, respectively. We say that $\mathfrak{M}$ is *linearly patched* if a linear order on its patches is provided. For a finite family $\mathcal{P}$ of relational structures, we say that $\mathfrak{M}$ is *linearly $\mathcal{P}$-patched*, if every patch of $\mathfrak{M}$ is isomorphic to some structure in $\mathcal{P}$.

In order to evaluate logical formulas on a linearly patched structure, we add the set of patches to its domain; additionally, we include a binary relation witch connects each patch with all its vertices, and also for each relation R of arity n we add a relation $\hat{R}$ which contains all tuples $(\mathfrak{p}, v_1, \ldots, v_n)$ consisting of a patch $\mathfrak{p}$ and n vertices, such that $R(v_1, \ldots, v_n)$ holds in $\mathfrak{p}$. This allows formulas to quantify both over vertices and patches. Finally, we allow formulas to refer to the linear order on patches.

Example 6.1. Let $\mathcal{P}$ be the singleton family containing the graph with two vertices and one undirected edge. Then a linearly $\mathcal{P}$-patched structure is the same thing as an undirected graph without isolated vertices, together with a linear order on its edges. As a purely relational structure, this is represented as the union of the set of vertices and the set of edges, together with a linear order on the set of edges, and a ternary relation which says that two given vertices are endpoints of a given edge.

Example 6.2. The CFI graphs [6] can be constructed from three-regular graphs, by replacing each vertex by a certain gadget $\mathfrak{G}$ with ten nodes, which correspond to the six atoms and four triples of the letter $\{(a,c,e),(a,d,f),(b,c,f),(b,d,e)\}$ from Example 2.4, and with edges connecting each triple with its elements. Neighboring gadgets are then connected by identifying two exchangeable pairs of atoms. If the original three-regular graph is ordered, then the resulting structure is a linearly $\mathcal{P}$-patched structure, where $\mathcal{P} = \{\mathfrak{G}\}$.

6.2 Linearly patched structures and words with atoms

For a relational structure $\mathfrak{M}$ consider the set $A_{\mathfrak{M}}$ of all relational structures isomorphic to $\mathfrak{M}$. Then $A_{\mathfrak{M}}$ is a single-orbit set.

Fact 6.1. *Every single-orbit set is related by an equivariant bijection to $A_{\mathfrak{M}}$, for some finite relational structure $\mathfrak{M}$.*

For a finite collection $\mathcal{P}$ of finite relational structures, let $A_{\mathcal{P}}$ denote the disjoint union of the alphabets $A_{\mathfrak{M}}$, for $\mathfrak{M} \in \mathcal{P}$.

Linearly $\mathcal{P}$-patched structures correspond to words over the alphabet $A_{\mathcal{P}}$. Indeed, a word w induces a linearly patched structure with $\sup(w)$ as the domain and the letters of w as patches. Conversely, a linearly $\mathcal{P}$-patched structure defines a word whose letters are the patches. This allows us to move freely between structures and words, and to view TMAs over $A_{\mathcal{P}}$ as recognisers of linearly $\mathcal{P}$-patched structures. For instance, by Fact 6.1, Theorem 3.1 may be reformulated as:

Lemma 6.2. *For every fixed finite collection $\mathcal{P}$, isomorphism of linearly $\mathcal{P}$-patched structures is recognizable by a deterministic TMA if and only if the alphabet $A_{\mathcal{P}}$ is standard.*

6.3 Order-invariant classical polynomial time

Let us come back to the world of classical (atomless) computations. Any relational structure has a *description*, i.e., a list of identifiers (one unique binary string per each element of the universe)

and for each relation, a list of its tuples, referred via the identifiers. Note that a fixed structure has many descriptions, depending on the chosen identifiers and orderings. In this non-unique way, relational structures are presented as inputs for classical Turing machines. We assume that these descriptions use a fixed finite alphabet, say $\{0, 1\}$.

A classical Turing machine is *order-invariant* if given on input a description of a relational structure, the output of the machine does not depend on the chosen description. For example, a Turing machine which checks whether a graph is connected is order-invariant, whereas a Turing machine which checks whether the first two vertices are adjacent is not.

Order-invariant Turing machines can be therefore viewed as recognizers of relational structures; however, it is undecidable whether a given Turing machine is order-invariant. One of the main open questions in Finite Model Theory, and in Descriptive Complexity Theory in particular, is whether there is a *logic which captures* order-invariant polynomial time. Such a logic is a class of formulas with decidable syntax, such that every formula can be effectively translated into a polynomial time order-invariant Turing machine accepting the same structures, and conversely: every property recognized by a polynomial time order-invariant Turing machine is definable by a formula. For example, first order logic satisfies only half of the requirements: a formula, such as $\forall x \exists y . E(x, y)$, gives rise to a polynomial time algorithm, which is order-invariant; however, graph connectedness is not definable by a first order formula.

One can relativize the above definitions to a class of structures $\mathcal{C}$, to say that a logic captures order-invariant polynomial time Turing machines *over* $\mathcal{C}$.

Observe that we can use two types of Turing machines to accept linearly $\mathcal{P}$-patched structures: deterministic TMAs, and classical Turing machines, which are order-invariant. The latter ones are – a priori – more powerful, as they have access to a linear ordering of the elements of the structure, which comes from the chosen description of the structure. On the other hand TMAs only have access to a linear ordering of the patches. Our results allow to characterize those families $\mathcal{P}$ for which the two types of machines are equally expressive. Furthermore, we relate these machines to the IFP logic.

6.4 IFP and IFP+C

We roughly describe the logics IFP and IFP+C. They are extensions of first order logic, which capture order-invariant polynomial time over some classes of structures, but not over all structures. For precise definitions and overviews of the cited results, see e.g. [11].

Instead of providing the precise definition of IFP, we give an illustrative example, and then roughly sketch the general form of the syntax.

Example 6.3. Consider structures over a relational language with one binary symbol E; they can be seen as directed graphs (possibly with self-loops). The following formula, with free variables x, y, says that there is a directed path from x to y:

$$\Big(\text{IFP}_{R,z}\big[(z = x) \vee \exists v : R(v) \wedge E(v, z)\big]\Big)(y) \qquad (\diamond)$$

The semantics is as follows. Let x, y be vertices of a graph. Start with R being the empty set of vertices, treated as a unary relation. Repeat indefinitely the following *inflationary* step:

> Add to R those vertices z which satisfy the subformula of ($\diamond$) delimited by the brackets [ ].

The formula ($\diamond$) holds for the pair of vertices (x, y) if y belongs to R in some step of the loop. Observe that the property described by ($\diamond$) is not first-order definable.

The general form of the construct is $\text{IFP}_{R,\bar{z}}[\phi(\bar{x}, \bar{z})](\bar{y})$. Comparing to ($\diamond$), there can be tuples of variables $\bar{x}, \bar{y}, \bar{z}$ instead of single variables x, y, z, under the restriction that $\bar{y}$ and $\bar{z}$ have the same length, say n. Then R is interpreted as an n-ary relation which is computed in an inflationary manner, starting from the empty relation.

The logic IFP+C further extends IFP by a construct $\#_{\bar{x}}\phi(\bar{x}, \bar{y})$ which allows to count (using a special counting sort) the number of assignments for $\bar{x}$ satisfying $\phi(\bar{x}, \bar{y})$.

It is not difficult to prove that a formula of IFP+C can be effectively translated into an equivalent polynomial time Turing machine, which is automatically order-invariant. The polynomial bound is a consequence of the fact that for finite models, due to the inflationary mode of computation, stabilization is reached after a polynomial number of steps.

The Immerman-Vardi theorem states that IFP (and, in consequence, IFP+C) captures order-invariant polynomial time over linearly ordered structures [12, 15]. On the other hand, IFP+C does not capture order-invariant polynomial time over all graphs, as proved in [6] using the CFI graphs. Our aim is to generalize both these results, using TMAs. How can TMAs be useful for studying the logic IFP?

IFP **over linearly $\mathcal{P}$-patched structures.** In the Immerman-Vardi theorem, instead of linearly ordered structures, one could equivalently use graphs (without isolated vertices) whose set of edges is linearly ordered. This is more consistent with linearly patched structures (see Example 6.1). The following proposition is then a generalization of the Immerman-Vardi theorem to linearly $\mathcal{P}$-patched structures instead of graphs with ordered edges, and TMAs instead of Turing machines.

Proposition 6.3. *The logics* IFP *and* IFP+C *capture polynomial time TMAs over linearly $\mathcal{P}$-patched structures.*

6.5 Main result

The main result of this section is:

Theorem 6.4. *The logic* IFP *captures order-invariant polynomial time over linearly $\mathcal{P}$-patched structures if and only if the alphabet $A_{\mathcal{P}}$ is standard.*

Observe that IFP can be replaced by IFP+C in the above theorem, since those logics are equivalent over linearly $\mathcal{P}$-patched structures by Proposition 6.3. Also, this proposition allows us to use polynomial time TMAs instead of the logic IFP in the proof.

Proof. Fix a finite family $\mathcal{P}$ of structures, and let $A_{\mathcal{P}}$ be the corresponding alphabet with atoms.

We first show the left-to-right implication. Assume that polynomial time deterministic TMAs capture classical order-invariant Turing machines over linearly $\mathcal{P}$-patched structures. Then, in particular, deterministic TMAs can express the isomorphism problem of linearly $\mathcal{P}$-patched structures, according to Lemma 6.5 below. Lemma 6.2 then implies that the alphabet $A_{\mathcal{P}}$ is standard, proving the left-to-right implication.

Lemma 6.5. *Isomorphism of linearly $\mathcal{P}$-patched structures is decidable by an order-invariant polynomial time Turing machine.*

Proof. As shown in Section 4, the isomorphism problem for linearly $\mathcal{P}$-patched structures reduces to the CSP problem over the template $T_{A_{\mathcal{P}}}$. For a fixed alphabet $A_{\mathcal{P}}$, the reduction is polynomial. It is known [5] that for a template with a Maltsev polymorphism, the induced CSP is in polynomial time. □

For the proof of the other implication, we show that if $A_{\mathcal{P}}$ is standard, then polynomial time TMAs are equally expressive as classical order-invariant polynomial time Turing machines.

Consider the language $D_{\mathcal{P}}$ containing all words of the form $w\#desc$, where $w \in A^*_{\mathcal{P}}$ and $desc \in \{0,1\}^*$ is a description (cf. Section 6.3) of the linearly $\mathcal{P}$-patched relational structure corresponding to w. Reusing the (k,l)-consistency algorithm for recognizing the language $D_{\mathcal{P}}$, we get the following:

Lemma 6.6. *If the alphabet $A_{\mathcal{P}}$ is standard, then the language $D_{\mathcal{P}}$ is recognized by a polynomial time TMA.*

The following lemma shows that if $A_{\mathcal{P}}$ is standard, then not only descriptions can be recognized, but also produced in polynomial time (i.e., there is a polynomial time *canonisation* algorithm).

Lemma 6.7. *Using a machine for the language $D_{\mathcal{P}}$ as blackbox, a polynomial time TMA can produce, for an input word $w \in A^*_{\mathcal{P}}$, a description $desc \in \{0,1\}^*$ of the structure corresponding to w.*

Proof. This can be done by finding descriptions of longer and longer prefixes of w. There is no need of backtracking, since any description of a prefix can be extended to a description of the whole word. Moreover, to extend the description of a prefix by one letter, the machine needs to check only constantly many possible candidate strings over $\{0,1\}$ and for each candidate, run the machine for $D_{\mathcal{P}}$ (the constant depends on the alphabet $A_{\mathcal{P}}$ but not on the length of the input). □

Suppose that $A_{\mathcal{P}}$ is standard and let L be a property of linearly $\mathcal{P}$-patched structures, recognized by a classical polynomial time Turing machine M. Then a TMA over $A_{\mathcal{P}}$ can decide, for an input word w, whether the structure corresponding to w has property L, in the following two steps. In the first step, the machine produces a description of the structure corresponding to the word w, according to Lemmas 6.6 and 6.7. In the second step the machine simulates, on that description, the order-invariant polynomial time Turing machine M recognizing L.

This proves the right-to-left implication of Theorem 6.4. □

Many results of the previous sections can be translated into results about logic. For example, from Section 5.1 and Theorem 6.4, we get:

Corollary 6.8. *If no structure in $\mathcal{P}$ has more than five elements, then* IFP *captures order-invariant polynomial time computations over linearly $\mathcal{P}$-patched structures.*

As a byproduct of the proof of Theorem 6.4 we obtain a "logic" that captures order-invariant polynomial time over linearly $\mathcal{P}$-patched structures (even if $A_{\mathcal{P}}$ is nonstandard), namely TMAs with an oracle for the language $D_{\mathcal{P}}$. This can be converted to a more reasonable logic (as in Proposition 6.3), namely, an extension of IFP+C by a construct which tests whether two linearly $\mathcal{P}$-patched structures are isomorphic.

7. Related work

The paper [1] also studies a relationship between the logic IFP+C and templates of bounded width. The emphasis there is to determine for which templates T there exists a formula of IFP+C which holds in a given structure (instance) I over the signature of T if and only if I maps homomorphically to T. It follows from that paper (see Corollary 23) and from later deep results from algebraic CSP theory [2] that this happens precisely when T has bounded width. In our paper, a similar, but weaker result is implicit (see Theorem 3.3, Proposition 4.3 and Proposition 6.3): it concerns only templates of the form T_A, where A is an alphabet. In particular, those templates have a Maltsev polymorphism.

Graph Isomorphism problem. The graph isomorphism problem with bounded color classes is the problem of deciding whether there is a color-preserving isomorphism between two given colored graphs G, H, where the coloring is such that at most k vertices get the same color (where k is a fixed parameter). This problem is the first restricted version of the graph isomorphism problem to be shown in PTime using group theoretic methods [9].

Colored graphs whose color classes have size at most k can be seen as patched structures, where the patches are the subgraphs induced by any pair of colors. Therefore, each patch is a graph with at most $2k$ vertices. Let $\mathcal{P}$ be the family of all graphs on vertices $\{1,\ldots,2k\}$. Then the isomorphism problem of such colored graphs reduces to the isomorphism problem for linearly $\mathcal{P}$-patched structures – the patches are linearly ordered according to an arbitrary linear ordering of the set of pairs of colors. It therefore follows from Lemma 6.5 that this can be done in polynomial time (and in logarithmic space if the alphabet $A_{\mathcal{P}}$ is standard, using [7]). Our proof does not rely on group theory, but instead uses the polynomial algorithm for solving CSPs over a template admitting a Maltsev polymorphism [5].

Acknowledgments

We are grateful to Marcin Kozik for guiding us in the land of Constraint Satisfaction Problems, to Mikołaj Bojańczyk for inspiring discussions on sets with atoms, and to anonymous reviewers for insightful comments.

References

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

[2] L. Barto and M. Kozik. Constraint satisfaction problems of bounded width. In *Procs. FOCS'09*, pages 595–603. IEEE Computer Society, 2009.

[3] M. Bojańczyk, B. Klin, and S. Lasota. Automata with group actions. In *Proc. LICS'11*, pages 355–364, 2011.

[4] M. Bojańczyk, B. Klin, S. Lasota, and S. Toruńczyk. Turing machines with atoms. In *Proc. LICS'13*, pages 183–192, 2013.

[5] A. A. Bulatov and V. Dalmau. A simple algorithm for Mal'tsev constraints. *SIAM J. Comput.*, 36(1):16–27, 2006.

[6] J. Cai, M. Fürer, and N. Immerman. An optimal lower bound on the number of variables for graph identifications. *Combinatorica*, 12(4): 389–410, 1992.

[7] V. Dalmau and B. Larose. Maltsev + Datalog –> symmetric Datalog. In *Procs. LICS*, pages 297–306. IEEE Computer Society, 2008.

[8] T. Feder and M. Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory. *SIAM J. Comput.*, 28(1):57–104, 1998.

[9] M. Furst, J. Hopcroft, and E. Luks. Polynomial-time algorithms for permutation groups. In *Proceedings of the 21st Annual Symposium on Foundations of Computer Science*, SFCS '80, pages 36–41. IEEE Computer Society, 1980.

[10] M. Gabbay and A. M. Pitts. A new approach to abstract syntax with variable binding. *Formal Asp. Comput.*, 13(3-5):341–363, 2002.

[11] M. Grohe. Descriptive complexity, canonisation, and definable graph structure theory, December 2013. URL `http://www.automata.rwth-aachen.de/~grohe/cap/index.en`.

[12] N. Immerman. Upper and lower bounds for first order expressibility. *J. Comput. Syst. Sci.*, 25(1):76–98, 1982.

[13] B. Larose, M. Valeriote, and L. Zádori. Omitting types, bounded width and the ability to count. *Int. J. Algebra and Computation*, 19(5):647–668, 2009.

[14] A. M. Pitts. *Nominal Sets: Names and Symmetry in Computer Science*. Cambridge University Press, 2013.

[15] M. Y. Vardi. The complexity of relational query languages (extended abstract). In *STOC*, pages 137–146, 1982.

A. Proofs for Section 3

A.1 Proof of Theorem 3.1 (sketch)

Since Iso_A is easily decidable by a nondeterministic TMA, the right-to-left implication is the only interesting one.

In [4] we considered providing TMAs with the *fresh oracle* that, whenever requested, returns an atom that does not appear in the current global configuration. A machine with the fresh oracle accepts a word if the oracle can respond so that the resulting run is accepting. By [4, Prop. VI.1], addition of the fresh oracle does not change the computing power of deterministic TMAs, for any input alphabet.

Consider a language L recognized by a *non*deterministic TMA M. A deterministic TMA that recognizes L may work as follows. Given an input word $w \in A^*$, use the fresh oracle multiple times, to generate as many fresh atoms as there are in the support of w. Write all these atoms on the tape, thus obtaining a total order on them. Then exhaustively generate, one by one, all words over A, of length equal to the length of w, built of the freshly generated atoms. (There are exponentially but finitely many such words). At least one of these words, say v, is isomorphic to w; find v using a deterministic test for the word isomorphic problem Iso_A that we assume to exist.

The machine M may then be run on the word v. However, now that the atoms in v are totally ordered, M may be simulated by a classical non-atomic computation, with every atom replaced by its number in the total order. The resulting machine can then be determinized using the classical backtracking construction.

A.2 Proof of Lemma 3.2

A deterministic TMA may compute the bags of a given w: first write down the support of every letter, and then calculate intersections and differences of these supports in an obvious manner. By writing down each bag as a work alphabet letter on its tape, our machine naturally computes the total ordering on bags as well. (Note that bags are of a bounded size, as each of them is contained in the support of a letter from A). Having computed the total orders of bags of v and w in this manner, checking whether v and w are similar is straightforward.

A.3 Proof of Theorem 3.3

One direction is immediate: if A has width (k, l), then the Turing machine computes $\mathit{cons}_{k,l}(vw)$ and accepts if and only if the resulting family is nonempty. To this end, it is enough to extend the work alphabet of the machine to include all possible translations of size at most l; this is possible since these families are of a bounded size.

For the other direction, we will need a few auxiliary notions.

Symbolic representations of orbits. We describe how the orbits of A^* can be designated in a way which does not use atoms. One way of describing an orbit of A^* is by providing any word w in this orbit; however, this representation uses atoms. Instead, we use symbolic representations of words.

A *symbolic representation* of a letter $a \in A$ is a word over a finite, atomless alphabet, for instance $\{0, 1\}$. It starts with some fixed identifier – a binary string – of the orbit of the letter a. Then comes a list of identifiers – binary strings – one for each element of the support of a. Additionally, the representation contains a list of automorphisms of the letter a; each automorphism is described as a permutation of identifiers to which it corresponds.

A *symbolic representation* of a word w consists of symbolic representations of each of its letters, with the requirement that if an atom appears in several letters, then it has the same identifier in the representations of these letters. Observe that a word w has several representations, depending on the chosen identifiers and their order. Below we use an ambiguous notation $\mathit{rep}(w)$ to denote *any* of the representations.

It is easy to see that two words $v, w \in A^*$ admit the same representation if and only if they are isomorphic. Moreover, there is a (polynomial time) nondeterministic TMA which given a word w on input, outputs its representation. As a result, the *representation language*

$$\mathit{Rep}_A = \{w\#\mathit{rep}(w) : \ w \in A^* \text{ and } \mathit{rep}(w) \text{ is a repr. of } w\}.$$

is recognizable by a polynomial time nondeterministic TMA.

Proof of Theorem 3.3, ctd. Suppose that the alphabet A is standard. Then the language Rep_A is recognized by a deterministic TMA. Let M be such a machine. Let k, l be two numbers depending on M, whose precise value we will specify later in the proof.

Let $v, w \in A^*$ be two words. Denote by $\mathcal{F}$ the family $\mathit{cons}_{k,l}(vw)$. We will show that if $\mathcal{F}$ is nonempty then v and w are isomorphic. This will imply that A has width (k, l).

The idea is as follows. Let $\mathit{rep}(v) \in \{0, 1\}^*$ be some representation of the word v. Then $v\#\mathit{rep}(v)$ is accepted by the machine M; let ρ be the accepting run. We will show that the family $\mathcal{F}$ can be used to translate ρ into an accepting run $\mathcal{F}(\rho)$ of M over the word $w\#\mathit{rep}(v)$. By the definition of Rep_A, this will imply that $\mathit{rep}(v)$ is a representation of the word w, so w and v are isomorphic.

Let x be an element whose support is contained in the support of v. Let y be some other element. For a translation σ we write f_σ for its union, seen as a partial bijection of atoms. We write $x \xrightarrow{\mathcal{F}} y$ if for every translation σ in $\mathcal{F}$, whenever f_σ is defined over the support of x then f_σ maps x to y, i.e. extends to a bijection of atoms which maps x to y.

We will apply the relation $\xrightarrow{\mathcal{F}}$ to single elements of configurations of M, such as states or tape letters. It will suffice that l is at least as big as the dimension of the work alphabet plus the dimension of the state space, and k at least as big as the maximum of the two dimensions. The relation $\xrightarrow{\mathcal{F}}$ will be also pointwise applied to whole configurations, or runs (sequences of configurations).

Lemma A.1. *Let ρ be a partial run of M over the word $v\#\mathit{rep}(v)$, let ρ' be a partial run of M over the word $w\#\mathit{rep}(v)$. Suppose that ρ and ρ' have the same length n. Then $\rho \xrightarrow{\mathcal{F}} \rho'$.*

Proof. By induction on n. The induction base follows immediately from the fact that $\mathcal{F}$ contains exclusively locally consistent translations. Note that $\mathit{rep}(v)$, being atomless, is preserved by translations.

For the inductive step, consider the last configurations of ρ and ρ', related by $\xrightarrow{\mathcal{F}}$ due to the induction assumption. In particular, the current head positions are necessarily exactly the same in both configurations. Let q, q' be the states of the configurations, and a, a' the tape contents at the head position. We will prove below that the next machine states are related by $\xrightarrow{\mathcal{F}}$. Similarly one argues for the values written to the tape, or head moves, which is sufficient to complete the induction step.

The next states p, p' of the machine M are the result of the equivariant transition function

$$p \ = \ \delta(q, a) \qquad p' \ = \ \delta(q', a').$$

In order to show $p \xrightarrow{\mathcal{F}} p'$ consider any translation $\sigma \in \mathcal{F}$ such that the domain of f_σ includes the support of p. It is enough to demonstrate that f_σ maps p to p', written shortly $\sigma(p) = p'$ below. Using downward closure of $\mathcal{F}$, restrict σ to a sub-translation of size at most k whose domain still includes the support of p (note that k is at least as big as the size of the support of p). Since $p = \delta(q, a)$, where δ is an equivariant function, the support of p is contained in the support of (q, a). Therefore, by weak upward closure of $\mathcal{F}$ we know that the restriction of σ extends to a translation $\bar{\sigma} \in \mathcal{F}$

whose domain includes the support of (q, a) (l is at least as big as the size of the support of q plus the size of the support of a). As by the induction assumption $(q, a) \xrightarrow{\mathcal{F}} (q', a')$, we know that $\bar{\sigma}(q, a) = (q', a')$. Hence, by equivariance of δ we deduce

$$\bar{\sigma}(p) \;=\; p'.$$

Knowing that σ and $\bar{\sigma}$ coincide on the support of p, we obtain

$$\sigma(p) \;=\; p'$$

as required. □

In particular, it follows that if there is an accepting run over $v\#\mathit{rep}(v)$ then the run of the same length over $w\#\mathit{rep}(v)$ is also accepting. Hence w and v are isomorphic. This finishes the proof of Theorem 3.3.

B. Proofs for Section 4

B.1 Proof of Lemma 4.1

Proof. In the special case when $v = w$ and $\alpha = \beta$, for each letter l the constraint relation R_l in the instance $I^{v,\alpha}_{v,\alpha}$ is a subgroup of $\mathcal{S}_{[B_1]} \times \cdots \times \mathcal{S}_{[B_n]}$. We will denote it by $G^{v,\alpha}_l$. In general, the constraint relations R_l which appear in the constraints of an instance $I^{v,\alpha}_{w,\beta}$ are cosets of $G^{v,\alpha}_l$ in the group $\mathcal{S}_{[B_1]} \times \cdots \times \mathcal{S}_{[B_n]}$, to be proved now.

Let l be a letter of v, and let $B_1, \ldots, B_n$ be its bags. For an atlas α we denote by α_l the tuple $(\alpha_{B_1}, \ldots, \alpha_{B_n})$. Every locally consistent translation σ with domain $\{B_1, \ldots, B_n\}$ induces a bijection f_σ from the disjoint union of B_i's to the disjoint union of B_i's. We identify this bijection with the tuple $(\sigma_{B_1}, \ldots, \sigma_{B_n})$. Let F_l be the set of all bijections of this form. In the special case when $v = w$, the set F_l is a subgroup of $\mathcal{S}_{B_1} \times \cdots \times \mathcal{S}_{B_n}$ where $\mathcal{S}_{B_i}$ is the symmetric group of all permutations of B_i. This group will be denoted G_l.

The tuples in R_l are in bijective correspondence with elements of F_l. More precisely each $\tau \in R_l$ is of the form $\tau = \alpha_l^{-1} \cdot (\sigma_{B_1}, \ldots, \sigma_{B_n}) \cdot \beta_l$, where $\{\sigma_{B_i} : i = 1, \ldots, n\}$ is locally consistent. Hence,

$$R_l = \alpha_l^{-1} \cdot F_l \cdot \beta_l.$$

Fix any $\sigma \in F_l$. Then $F_l = G_l \cdot \sigma$. Therefore,

$$R_l = \alpha_l^{-1} \cdot G_l \cdot \sigma \cdot \beta_l = \alpha_l^{-1} \cdot G_l \cdot \alpha_l \cdot \alpha_l^{-1} \cdot \sigma \cdot \beta_l.$$

Since $G^{v,\alpha}_l = \alpha_l^{-1} \cdot G_l \cdot \alpha_l$ is a subgroup of $\mathcal{S}_{[B_1]} \times \cdots \times \mathcal{S}_{[B_n]}$, we have that $R_l = G^{v,\alpha}_l \cdot \tau$, where $\tau = \alpha_l^{-1} \cdot \sigma \cdot \beta_l$. Thus, R_l is a coset in $\mathcal{S}_{[B_1]} \times \cdots \times \mathcal{S}_{[B_n]}$. If $v = w$ and $\alpha = \beta$ this gives us $R_l = G^{v,\alpha}_l$. □

B.2 Proof of Proposition 4.3

One implication of Proposition 4.3, namely from (2) to (1), is immediate from Fact 4.2. We concentrate now on the converse implication. We need some auxiliary notions.

Homomorphically equivalent instances. It is convenient to think of an instance I over a template T, as a relational structure over the vocabulary of all relations R of T. The elements of the relational structure are the variables of the instance. Every constraint $C = (\bar{x}, R)$ in I is then understood as the formal relation R relating the tuple $\bar{x}$ of elements of the relational structure. More importantly, (partial) solutions to I coincide with (partial) homomorphisms from the relational structure to the template. This setting allows us to speak of *homomorphically equivalent* instances over the same template, i.e., instances I, I' admitting two homomorphisms

$$h : I \to I' \qquad \text{and} \qquad h' : I' \to I \tag{6}$$

of relational structures.

The implication from (1) to (2) of Proposition 4.3 is shown using the following lemma:

Lemma B.1. *Let I be an instance over the template T_A, such that $\mathit{cons}_{k,l}(I) \neq \emptyset$. There exist words $v, w \in A^*$ and their atlases α, β, for which the instance $I' = I^{v,\alpha}_{w,\beta}$ is homomorphically equivalent to I.*

Indeed, suppose $\mathit{cons}_{k,l}(I) \neq \emptyset$ for an instance I over T_A. Due to the lemma one obtains an instance $I' = I^{v,\alpha}_{w,\beta}$, and a homomorphism $h' : I' \to I$ that allows us to translate $\mathit{cons}_{k,l}(I)$ to a nonempty (k, l)-consistent family of partial solutions to I', by pre-composing with h'. Since the alphabet A has width (k, l), this family enforces the existence of a solution to I', which is then translated back, by pre-composing with a homomorphism $h : I \to I'$, to a solution to I, as required.

It remains to show Lemma B.1. We call an instance I *distinguishing*, if:

- there are no non-constrained variables in I, i.e., every variable appears in some constraint of I,
- for every two variables y, z of I, there is a constraint $C = (\bar{x}, R)$ such that exactly one of the variables y, z appears in the constrained tuple $\bar{x}$ of variables.

We say that the constraint C in the definition above *distinguishes* y and z. For instance, every unary constraint (x, R) distinguishes x from any other variable.

Lemma B.1 follows immediately from the following two facts:

Claim B.1.1. *For every instance I over some template T, there is a distinguishing instance I_d over the same template homomorphically equivalent to I.*

Claim B.1.2. *Let I be a distinguishing instance over the template T_A, such that $\mathit{cons}_{k,l}(I) \neq \emptyset$. There exist words $v, w \in A^*$ and their atlases α, β, for which the instance $I' = I^{v,\alpha}_{w,\beta}$ is isomorphic (as a relational structure) to I.*

Proof of Lemma B.1. Starting with an instance I over T_A such that $\mathit{cons}_{k,l}(I) \neq \emptyset$, using Claim B.1.1 one gets a distinguishing instance I_d. Similarly as before, one uses a homomorphism $h_d : I_d \to I$ to show that $\mathit{cons}_{k,l}(I_d) \neq \emptyset$. This allows us to apply Claim B.1.2 to I_d, in order to derive v, w, α and β as required. □

Proof of Claim B.1.1. Let I be an arbitrary instance over some template T. We define a distinguishing instance I_d as follows. All the variables which appear in any constraint of I are variables of I_d. Additionally, for every variable y of I that is constrained by a non-unary constraint $C = (\bar{x}, R)$ of I, i.e. such that y appears in $\bar{x} = (x_1, \ldots, x_n)$ and $n > 1$, add to I_d a *primed* variable y'. All the constraints of I are constraints of I_d. Additionally, for every non-unary constraint $C = (\bar{x}, R)$ as above, add n new constraints to I_d, each obtained by replacing one of the variables x_i in C by its primed version. For instance, for $n = 3$, the following three new constraints are added to I_d:

$$((x'_1, x_2, x_3), R)$$
$$((x_1, x'_2, x_3), R)$$
$$((x_1, x_2, x'_3), R).$$

It remains to verify that I_d is distinguishing. The first condition is obviously satisfied. It is also easy to see that every two variables, being primed or not, are distinguished by some constraint. For instance, the constraint $((x'_1, x_2, x_3), R)$ distinguishes x_1 and x_2, and also x_1 and x'_1, but does not distinguish x_2 and x_3. As another example, the original constraint $((x_1, x_2, x_3), R)$ of I distinguishes x_1 and y', for any primed variable y'.

If there are no non-constrained variables in I, then I is a subinstance of I_d, and the embedding is a homomorphism. If there are any non-constrained variables the homomorphism can map them to arbitrarily chosen variables of I_d. For the other direction, the function that maps every primed variable y' to y, and preserves all non-primed variables, is a homomorphism from I_d to I. Thus I and I_d are homomorphically equivalent. □

Proof of Claim B.1.2. Given a distinguishing instance I over the template T_A, we construct similar words v, w and their atlases α, β.

Recall from Section 4.1 that every relation R appearing in some constraint $C = ((x_1, \ldots, x_n), R)$ of I is a coset in a group of the form $\mathcal{S}_{[k_1]} \times \cdots \times \mathcal{S}_{[k_n]}$. Thus the constraint associates, to every variable x_i, the cardinality k_i. We can assume that the cardinalities associated to a variable by different constraints are the same, as otherwise $cons_{k,l}(I) = \emptyset$, contrary to the assumption.

Choose, for every variable x of I, some sets B_x and $\tilde{B}_x$ of atoms, of size equal to the cardinality k_x associated to x. We assume that the sets B_x and $\tilde{B}_x$ are pairwise disjoint. The sets B_x will be the bags in the word v, and the sets $\tilde{B}_x$ will be the bags in the word w. Furthermore, choose some arbitrary maps:

$$\alpha_x : B_x \to [B_x] = [k_x]$$
$$\beta_x : \tilde{B}_x \to [\tilde{B}_x] = [k_x].$$

The maps will form atlases of v and w, respectively. Note that any permutation τ of $[k_x]$ induces a bijection σ from B_x to $\tilde{B}_x$:

$$\sigma = \alpha_x \cdot \tau \cdot (\beta_x)^{-1}. \tag{7}$$

In order to build the words v and w, we start with the empty words and consider sequentially all the constraints of I. For every constraint C, we append one letter l_C to v and one letter $\tilde{l}_C$ to w. Fix a constraint $C = ((x_1, \ldots, x_n), R)$. Note that $R \subseteq \mathcal{S}_{[k_1]} \times \cdots \times \mathcal{S}_{[k_n]}$. Define l_C and $\tilde{l}_C$ as two arbitrary letters satisfying

$$\sup(l_C) = B_{x_1} \cup \ldots \cup B_{x_n}$$
$$\sup(\tilde{l}_C) = \tilde{B}_{x_1} \cup \ldots \cup \tilde{B}_{x_n}$$

and such that the set of those tuples $(\sigma_1, \ldots, \sigma_n)$ that induce an isomorphism from l_C to $\tilde{l}_C$ corresponds bijectively, via (7), to the set of tuples $(\tau_1, \ldots, \tau_n) \in R$. Note that by the construction in Section 4.1 such letters $l_C, \tilde{l}_C$ exist for every relation R in the template T_A. This ends the construction of the words v, w. The atlases α, β are derived from the maps α_x and β_x.

We claim that the bags of v are exactly the sets B_x. Indeed, by the assumption that I is distinguishing, for every variable x there is a letter of v that contains B_x. Moreover, for every two variables x, y there is a letter of v whose support contains exactly one of the sets B_x, B_y and is disjoint from the other. Thus every set B_x is uniquely determined by the sequence of letters which contain it. Similarly, the bags of w are exactly the sets $\tilde{B}_x$. Moreover, the bag $\tilde{B}_x$ corresponds to the bag B_x.

We now argue that I is isomorphic, as a relational structure, to the instance $I' = I^{v,\alpha}_{w,\beta}$. Indeed, there is a one-to-one correspondence between variables of I and variables of I' given by $x \mapsto B_x$. Moreover, for any constraint $C = ((x_1, \ldots, x_n), R)$ of I there is a corresponding letter l_C in v with bags $B_{x_1}, \ldots, B_{x_n}$ (the correspondence is again bijective). Finally, the constraints of the instance I' correspond to letters l_C, and it follows from the construction that the tuple $(B_{x_1}, \ldots, B_{x_n})$ is constrained to the relation R. □

B.3 Proof of Lemma 4.5

Let C be a core of T_A, and let r be a homomorphism of T_A onto C which is the identity on C. If the template T_A has a majority polymorphism m then the function $r \circ m$ is a majority polymorphism of the core C.

For the opposite implication let m be a majority polymorphism of C. We define a function $m' : D^3 \to D$. For any $x, y \in D$ let $m'(x, x, y) = m'(x, y, x) = m'(y, x, x) = x$ and if $x, y, z \in D$ are pairwise distinct put $m'(x, y, z) = m(r(x), r(y), r(z))$.

Relations in the template T_A are cosets. Let $\mathcal{R}$ be a set of relations which are cosets of some fixed subgroup G in a group $\mathcal{S}_{[n_1]} \times \cdots \times \mathcal{S}_{[n_k]}$. The set $\mathcal{R}$ forms a partition of $\mathcal{S}_{[n_1]} \times \cdots \times \mathcal{S}_{[n_k]}$. Take any $R \in \mathcal{R}$ and consider three tuples $\bar{x}, \bar{y}, \bar{z}$ that belong to R. Observe that $m'(\bar{x}, \bar{y}, \bar{z}) \in \mathcal{S}_{[n_1]} \times \cdots \times \mathcal{S}_{[n_k]}$ and therefore, $m'(\bar{x}, \bar{y}, \bar{z}) \in R'$ for some $R' \in \mathcal{R}$. Suppose that $R' \neq R$. We have that $r(m'(\bar{x}, \bar{y}, \bar{z})) \in R'$. On the other hand, $r(m'(\bar{x}, \bar{y}, \bar{z})) = m(r(\bar{x}), r(\bar{y}), r(\bar{z}))$. Hence, $r(m'(\bar{x}, \bar{y}, \bar{z})) \in R$ and we obtain a contradiction since R and R' are disjoint.

C. Proofs for Section 6

C.1 Proof of Proposition 6.3

For one direction, we show how an IFP sentence ϕ can be converted into a TMA, by induction on the size of ϕ. To proceed with the induction, we describe how a TMA can simulate a formula ϕ with free variables. For convenience, instead of linearly $\mathcal{P}$-patched structures, we speak of words over $A_{\mathcal{P}}$ (cf. Subsection 6.2).

Let $\phi(\bar{x}, \bar{y})$ be a formula. This notation indicates that $\bar{x} = (x_1, \ldots, x_k)$ is a tuple of variables ranging over positions, and that $\bar{y} = (y_1, \ldots, y_l)$ is a tuple of variables ranging over atoms. Let $w = a_1 \ldots a_n$ be an input word. By $\phi[w]$ we denote the $(k+l)$-ary function which associates to a $(k+l)$-tuple $(i_1, \ldots, i_k, j_1, \ldots, j_l)$ of positions of w, the set of those elements $\bar{u} \in \sup(a_{j_1}) \times \cdots \times \sup(a_{j_l})$ such that $w, \bar{i}, \bar{u} \models \phi(\bar{x}, \bar{y})$.

In particular, if ϕ is a sentence, then $\phi[w]$ is a 0-ary function – a constant – which is either the empty set (if $w \not\models \phi$) or the singleton consisting of the empty tuple (if $w \models \phi$).

Lemma C.1. *Let $\phi(\bar{x}, \bar{y})$ be a formula of* IFP. *Then there exists a polynomial time TMA M_ϕ which, for an input word w, computes the function $\phi[w]$.*

Proof. Easy induction on the size of ϕ. □

We now proceed to the second part of the proof of Proposition 6.3. Assume that M is a polynomial time TMA, and let p be a polynomial bounding its running time. Let k be an integer such that $p(x) \leq x^k$ for $x \geq 2$. Let B be the work alphabet and Q be the state space of the machine M. Without loss of generality (c.f. Fact 6.1), we assume that the sets B and Q are of the form $A_{\mathcal{R}}$, for some finite family of structures $\mathcal{R}$.

The following lemma is useful for showing that a single step of the computation of a deterministic TMA can be simulated by a first order formula.

Lemma C.2. *Let $\mathfrak{M}_0$ and $\mathfrak{N}_0$ be two finite relational structures. Let $f : A_{\mathfrak{M}_0} \to A_{\mathfrak{N}_0}$ be an equivariant function. Then, for any structure $\mathfrak{M} \in A_{\mathfrak{M}_0}$, the structure $f(\mathfrak{M})$ interprets in $\mathfrak{M}_0$ via a first order interpretation which depends only on f.*

Proof. Recall that for any equivariant function f, the result $f(\mathfrak{M})$ is supported by the support of $\mathfrak{M}$. In this case, this means that the domain of the structure $f(\mathfrak{M})$ is contained in the domain of the structure $\mathfrak{M}$. As a consequence, if R is an n-ary relation of the structure $f(\mathfrak{M})$, then R is a subset of $(\sup(\mathfrak{M}))^n$. Moreover, R is invariant under the automorphisms of $\mathfrak{M}$: if $\pi \in \mathrm{Aut}(\mathfrak{M})$, then π extends to a bijection of atoms which fixes $\mathfrak{M}$, so it fixes $f(\mathfrak{M})$, and therefore π fixes the relation R.

To finish the proof, observe that an n-ary relation R in a finite structure $\mathfrak{M}$ is invariant under the automorphisms of $\mathfrak{M}$ if and only if R is first-order definable. □

Since the alphabets Q and B are finite unions of alphabets of the form $A_{\mathfrak{M}}$, we get the following.

Corollary C.3. *The transition function δ of the TMA M is first-order definable.*

This allows us to simulate one step of the computation of a TMA by a first-order formula. To simulate the entire run, we use IFP, as in the standard proof of the Immerman-Vardi theorem (see e.g. [11]).

Algebraic Properties of Valued Constraint Satisfaction Problem

Marcin Kozik[1] and Joanna Ochremiak[2,⋆]

[1] Jagiellonian University
[2] University of Warsaw

Abstract. The paper presents an algebraic framework for optimization problems expressible as Valued Constraint Satisfaction Problems. Our results generalize the algebraic framework for the decision version (CSPs) provided by Bulatov et al. [SICOMP 2005].
We introduce the notions of weighted algebras and varieties, and use the Galois connection due to Cohen et al. [SICOMP 2013] to link VCSP languages to weighted algebras. We show that the difficulty of VCSP depends only on the weighted variety generated by the associated weighted algebra.
Paralleling the results for CSPs we exhibit a reduction to cores and rigid cores which allows us to focus on idempotent weighted varieties. Further, we propose an analogue of the Algebraic CSP Dichotomy Conjecture; prove the hardness direction and verify that it agrees with known results for VCSPs on two-element sets [Cohen et al. 2006], finite-valued VCSPs [Thapper and Živný 2013], and conservative VCSPs [Kolmogorov and Živný 2013].

1 Introduction

An instance of the Constraint Satisfaction Problem (CSP) consists of variables (to be evaluated in a domain) and constraints restricting the evaluations. The aim is to find an evaluation satisfying all the constraints or satisfying the maximal possible number of constraints or approximating the maximal possible number of satisfied constraints etc. depending on the version of the problem. Further one can divide constraint satisfaction problems with respect to the size of the domain, the allowed constraints or the shape of the instances.

A particularly interesting version of the CSP was proposed in a seminal paper of Feder and Vardi [11]. In this version the CSP is defined by a *language* which consists of relations over a finite set. An instance of such a CSP is allowed if all the constraint relations are from this set. The goal is to determine whether an instance has a solution satisfying all the constraints.

⋆ The first author was supported by the Polish National Science Centre (NCN) grant 2011/01/B/ST6/01006; the second author was supported by the Polish National Science Centre (NCN) grant 2012/07/B/ST6/01497.

Each language clearly defines a problem in NP; the whole family of problems is interesting for another reason: it is robust enough to include some well studied computational problems, e.g. 2-colorability, 3-SAT, solving systems of linear equations over $\mathbb{Z}_p$, and still is conjectured [11] not to contain problems of intermediate complexity. This conjecture (which holds for languages on two-element sets by the result of Schaefer [19]) is known as the Constraint Satisfaction Dichotomy Conjecture of Feder and Vardi. Confirming this conjecture would establish CSPs as one of the largest natural subclasses of NP without problems of intermediate complexity.

The conjecture always attracted a lot of attention, but the first results, even very interesting ones, were usually very specialized (e.g. [12]). A major breakthrough appeared with a series of papers establishing *the algebraic approach to CSP* [3, 7, 14]. This deep connection with an independently developed branch of mathematics introduced a new viewpoint and provided tools necessary to tackle wide classes of CSP languages at once. At the heart of this approach lies a Galois connection between languages and clones of operations called *polymorphisms* (which completely determine the complexity of the language).

Results obtained using these new methods include a full complexity classifications for CSPs on three-element sets [5] and those containing all unary relations [4, 6]. Moreover, the algebraic approach to CSP allowed to propose a boundary between the tractable and NP-complete problems: this conjecture is known as the Algebraic Dichotomy Conjecture. Unfortunately, despite many efforts (e.g. [5]), both conjectures remain open.

The Valued Constraint Satisfaction Problem (VCSP) further extends the approach proposed by Feder and Vardi. The role of constraints is played by *cost functions* describing the price of choosing particular values for variables as a part of the solution. This generalization allows to construct languages modeling standard optimization problems, for example MAX-CUT. Moreover, by allowing ∞ as a cost of a tuple, a VCSP language can additionally model every problem that CSP can model, as well as hybrid problems like MIN-VERTEX-COVER. This makes the extended framework even more general (compare the survey [15]).

A number of classes of VCSPs have been thoroughly investigated. The underlying structure suggested capturing the properties of languages of cost functions using an amalgamation of algebraic and numerical techniques [10, 22]. The first approach which provides a Galois correspondence (mirroring the Galois correspondence for CSPs) was proposed by Cohen et al. [9]. A weighted clone defined in this paper fully captures the complexity of a VCSP language.

The present paper builds on that correspondence imitating the line of research for CSPs [7]. It is organized in the following way: Section 2 contains preliminaries and basic definitions. In Section 3 we present a reduction to cores and rigid cores. Section 4 introduces a concept of a weighted algebra and a weighted variety, and shows that those notions are well behaved in the context of the Galois connection for VCSP. Reductions developed in Section 3 together with definitions from Section 4 allow us to focus on idempotent varieties. Section 5 states a conjecture postulating (for idempotent varieties) the division

between the tractable and NP-hard cases of VCSP. The conjecture is clearly a strengthening of the Algebraic Dichotomy Conjecture [7]. Section 5 contains additionally the proof of the hardness direction of the conjecture as well as the reasoning showing that the conjecture agrees with complexity classifications for VCSPs on two-element sets [10], with finite-valued cost functions [22], and with conservative cost functions [17].

2 Preliminaries

2.1 The Valued Constraint Satisfaction Problem

Throughout the paper, let $\overline{\mathbb{Q}} = \mathbb{Q} \cup \{\infty\}$. We assume that $x + \infty = \infty$ and $y \cdot \infty = \infty$ for $y \geq 0$. An r-ary *relation* on a set D is a subset of D^r, a *cost function* on D of arity r is a function from D^r to $\overline{\mathbb{Q}}$. We denote by Φ_D the set of all cost functions on D. A cost function which takes only finite values is called *finite-valued*. A $\{0, \infty\}$-valued cost function is called *crisp* and can be viewed as a relation.

Definition 1. *An* instance of the valued constraint satisfaction problem (VCSP) *is a triple $\mathcal{I} = (V, D, \mathcal{C})$ with V a finite set of* variables*, D a finite* domain *and $\mathcal{C}$ a finite multi-set of* constraints*. Each constraint is a pair $C = (\sigma, \varrho)$ with σ a tuple of variables of length r and ϱ a cost function on D of arity r.*

An assignment *for $\mathcal{I}$ is a mapping $s\colon V \to D$. The* cost *of an assignment s is given by $Cost_{\mathcal{I}}(s) = \sum_{(\sigma,\varrho)\in\mathcal{C}} \varrho(s(\sigma))$ (where s is applied component-wise). To solve $\mathcal{I}$ is to find an assignment with a minimal cost, called an* optimal *assignment.*

Any set $\Gamma \subseteq \Phi_D$ is called a *valued constraint language* over D, or simply a *language*. If all cost functions from Γ are $\{0, \infty\}$-valued or finite-valued, we call it a *crisp* or *finite-valued* language, respectively.

By VCSP(Γ) we denote the class of all VSCP instances in which all cost functions in all constraints belong to Γ. VCSP(Γ_{crisp}), where Γ_{crisp} is the language consisting of all crisp cost functions on some fixed set D, is equivalent to the classical CSP. For an instance $\mathcal{I} \in$ VCSP(Γ) we denote by $\mathrm{Opt}_\Gamma(\mathcal{I})$ the cost of an optimal assignment. We say that a language Γ is *tractable* if, for every finite subset $\Gamma' \subseteq \Gamma$, there exists an algorithm solving any instance $\mathcal{I} \in$ VCSP(Γ') in polynomial time, and we say that Γ is *NP-hard* if VCSP(Γ') is NP-hard for some finite $\Gamma' \subseteq \Gamma$.

Weighted Relational Clones. We follow the exposition of [9] and define a closure operator on valued constraint languages that preserves tractability.

Definition 2. *A cost function ϱ is* expressible *over a valued constraint language $\Gamma \subseteq \Phi_D$ if there exists an instance $\mathcal{I}_\varrho \in$ VCSP(Γ) and a list $(v_1, \ldots, v_r)$ of variables of $\mathcal{I}_\varrho$, such that*

$$\varrho(x_1, \ldots, x_r) = \min_{\{s\colon V\to D \;\mid\; s(v_i)=x_i\}} Cost_{\mathcal{I}_\varrho}(s).$$

Note that the list of variables $(v_1, \ldots, v_r)$ in the definition above might contain repeated entries. Hence, it is possible that there are no assignments s such that $s(v_i) = x_i$ for all i. We define the minimum over the empty set to be ∞.

Definition 3. *A set $\Gamma \subseteq \Phi_D$ is a* weighted relational clone *if it is closed under expressibility, scaling by non-negative rational constants, and addition of rational constants. We define* $\mathrm{wRelClo}(\Gamma)$ *to be the smallest weighted relational clone containing Γ.*

If $\varrho(x_1, \ldots, x_r) = \varrho_1(y_1, \ldots, y_s) + \varrho_2(z_1, \ldots, z_t)$ for some fixed choice of arguments $y_1, \ldots, y_s, z_1, \ldots, z_t$ from amongst $x_1, \ldots, x_r$ then the cost function ϱ is said to be obtained by *addition* from the cost functions ϱ_1 and ϱ_2. It is easy to see that a weighted relational clone is closed under addition, and minimisation over arbitrary arguments.

The following result shows that we can restrict our attention to languages which are weighted relational clones.

Theorem 4 (Cohen et al. [9]). *A valued constraint language Γ is tractable if and only if* $\mathrm{wRelClo}(\Gamma)$ *is tractable, and it is NP-hard if and only if* $\mathrm{wRelClo}(\Gamma)$ *is NP-hard.*

Weighted polymorphisms. A k-ary *operation* on D is a function $f\colon D^k \to D$. We denote by $\mathcal{O}_D$ the set of all finitary operations on D and by $\mathcal{O}_D^{(k)}$ the set of all k-ary operations on D. The k-ary *projections*, defined for all $i \in \{1, \ldots, k\}$, are the operations $\pi_i^{(k)}$ such that $\pi_i^{(k)}(x_1, \ldots, x_k) = x_i$. Let $f \in \mathcal{O}_D^{(k)}$ and $g_1, \ldots, g_k \in \mathcal{O}_D^{(l)}$. The l-ary operation $f[g_1, \ldots, g_k]$ defined by $f[g_1, \ldots, g_k](x_1, \ldots, x_l) = f(g_1(x_1, \ldots, x_l), \ldots, g_k(x_1, \ldots, x_l))$ is called the *superposition* of f and $g_1, \ldots, g_k$.

A set $C \subseteq \mathcal{O}_D$ is a *clone of operations* (or simply a *clone*) if it contains all projections on D and is closed under superposition. The set of k-ary operations in a clone C is denoted $C^{(k)}$. The smallest possible clone of operations over a fixed set D is the set of all projections on D, which we denote Π_D.

Following [9] we define a k-ary *weighting* of a clone C to be a function $\omega\colon C^{(k)} \to \mathbb{Q}$ such that $\sum_{f \in C^{(k)}} \omega(f) = 0$, and if $\omega(f) < 0$ then f is a projection. The set of operations to which a weighting ω assigns positive weights is called the *support* of ω and denoted $\mathrm{supp}(\omega)$.

A new weighting of the same clone can be obtained by scaling a weighting by a non-negative rational, adding two weightings of the same arity and by the following operation called *superposition*.

Definition 5. *Let ω be a k-ary weighting of a clone C and let $g_1, \ldots, g_k \in C^{(l)}$. A* superposition *of ω and $g_1, \ldots, g_k$ is a function $\omega[g_1, \ldots, g_k]\colon C^{(l)} \to \mathbb{Q}$ defined by*

$$\omega[g_1, \ldots, g_k](f') \;=\; \sum_{\{f \in C^{(k)} \;\mid\; f[g_1, \ldots, g_k] = f'\}} \omega(f).$$

The sum of weights that any superposition $\omega[g_1, \ldots, g_k]$ assigns to the operations in $C^{(l)}$ is equal to zero, however, it may happen that a superposition assigns a negative value to an operation that is not a projection. A superposition is said to be *proper* if the result is a valid weighting.

A non-empty set of weightings over a fixed clone C is called a *weighted clone* if it is closed under non-negative scaling, addition of weightings of equal arity and proper superposition with operations from C. For any clone of operations C, the set of all weightings over C and the set of all zero-valued weightings of C are weighted clones.

We say that an r-ary relation R on D is *compatible* with an operation $f\colon D^k \to D$ if, for any list of r-tuples $\mathbf{x_1}, \ldots, \mathbf{x_k} \in R$ we have $f(\mathbf{x_1}, \ldots, \mathbf{x_k}) \in R$ (where f is applied coordinate-wise). Let $\varrho\colon D^r \to \overline{\mathbb{Q}}$ be a cost function. We define $\mathrm{Feas}(\varrho) = \{\mathbf{x} \in D^r \mid \varrho(\mathbf{x}) \text{ is finite}\}$ to be the *feasibility relation* of ϱ. We call an operation $f\colon D^k \to D$ a *polymorphism* of ϱ if the relation $\mathrm{Feas}(\varrho)$ is compatible with it. For a valued constraint language Γ we denote by $\mathrm{Pol}(\Gamma)$ the set of operations which are polymorphisms of all cost functions $\varrho \in \Gamma$. It is easy to verify that $\mathrm{Pol}(\Gamma)$ is a clone. The set of m-ary operations in $\mathrm{Pol}(\Gamma)$ is denoted $\mathrm{Pol}_m(\Gamma)$.

For crisp cost functions (relations) this notion of polymorphism corresponds precisely to the standard notion of polymorphism which has played a crucial role in the complexity analysis for the CSP [3, 14].

Definition 6. *Take ϱ to be a cost function of arity r on D, and let $C \subseteq \mathrm{Pol}(\{\varrho\})$ be a clone of operations. A weighting $\omega\colon C^{(k)} \to \mathbb{Q}$ is called a* weighted polymorphism *of ϱ if, for any list of r-tuples $\mathbf{x_1}, \ldots, \mathbf{x_k} \in \mathrm{Feas}(\varrho)$, we have*

$$\sum_{f \in C^{(k)}} \omega(f) \cdot \varrho(f(\mathbf{x_1}, \ldots, \mathbf{x_k})) \leq 0.$$

For a valued constraint language Γ we denote by $\mathrm{wPol}(\Gamma)$ the set of those weightings of the clone $\mathrm{Pol}(\Gamma)$ that are weighted polymorphisms of all cost functions $\varrho \in \Gamma$. The set of weightings $\mathrm{wPol}(\Gamma)$ is a weighted clone [9].

An operation f is *idempotent* if $f(x, ..., x) = x$. A weighted polymorphism is called *idempotent* if all operations in its support are idempotent. An operation $f \in \mathcal{O}_D^{(k)}$ is *cyclic* if for every $x_1, \ldots, x_k \in D$ we have that $f(x_1, x_2, \ldots, x_k) = f(x_2, \ldots, x_k, x_1)$. A weighted polymorphism is called *cyclic* if its support is non-empty and contains cyclic operations only.

A cost function ϱ is said to be *improved* by a weighting ω if ω is a weighted polymorphism of ϱ. For any set W of weightings over a fixed clone $C \subseteq \mathcal{O}_D$ we denote by $\mathrm{Imp}(W)$ the set of cost functions on D which are improved by all weightings $\omega \in W$.

By the result of Cohen et al. [9] for any finite valued constraint language Γ, we have $\mathrm{Imp}(\mathrm{wPol}(\Gamma)) = \mathrm{wRelClo}(\Gamma)$. This fact, together with Theorem 4, implies that tractable valued constraint languages can be characterized by their weighted polymorphisms.

2.2 Algebras and varieties

An *algebraic signature* is a set of function symbols together with (finite) arities. An *algebra* $\mathbf{A}$ over a fixed signature Σ, has a *universe* A, and a set of *basic operations* that correspond to the symbols in the signature, i.e., if the signature contains a k-ary symbol f then the algebra has a basic operation $f^{\mathbf{A}}$, which is a function $f^{\mathbf{A}}\colon A^k \to A$.

A subset B of the universe of an algebra $\mathbf{A}$ is a *subuniverse* of $\mathbf{A}$ if it is closed under all operations of $\mathbf{A}$. An algebra $\mathbf{B}$ is a *subalgebra* of $\mathbf{A}$ if B is a subuniverse of $\mathbf{A}$ and the operations of $\mathbf{B}$ are restrictions of all the operations of $\mathbf{A}$ to B. Let $(\mathbf{A}_i)_{i\in I}$ be a family of algebras (over the same signature). Their *product* $\Pi_{i\in I}\mathbf{A}_i$ is an algebra with the universe equal to the cartesian product of the A_i's and operations computed coordinate-wise. For two algebras $\mathbf{A}$ and $\mathbf{B}$ (over the same signature), a *homomorphism* from $\mathbf{A}$ to $\mathbf{B}$ is a function $h\colon A \to B$ that preserves all operations. It is easy to see, that an image of an algebra under a homomorphism $h\colon A \to B$ is a subalgebra of $\mathbf{B}$.

Let $\mathcal{K}$ be a class of algebras over a fixed signature Σ. We denote by $\mathrm{S}(\mathcal{K})$ the class of all subalgebras of algebras in $\mathcal{K}$, by $\mathrm{P}(\mathcal{K})$ the class of all products of algebras in $\mathcal{K}$, by $\mathrm{P}_{fin}(\mathcal{K})$ the class of all finite products, and by $\mathrm{H}(\mathcal{K})$ the class of all homomorphic images of algebras in $\mathcal{K}$. If $\mathcal{K} = \{\mathbf{A}\}$ we write $\mathrm{S}(\mathbf{A})$, $\mathrm{P}(\mathbf{A})$, and $\mathrm{H}(\mathbf{A})$ instead of $\mathrm{S}(\{\mathbf{A}\})$, $\mathrm{P}(\{\mathbf{A}\})$, and $\mathrm{H}(\{\mathbf{A}\})$, respectively.

A *variety* $\mathcal{V}(\mathcal{K})$ is the smallest class of algebras closed under all three operations. For an algebra $\mathbf{A}$ the variety $\mathcal{V}(\{\mathbf{A}\})$ (denoted $\mathcal{V}(\mathbf{A})$) is the variety *generated* by $\mathbf{A}$, and $\mathcal{V}_{fin}(\mathbf{A})$ is the class of finite algebras in $\mathcal{V}(\mathbf{A})$. Due to a result of Tarski [20] we know that for any finite algebra $\mathbf{A}$, we have

$$\mathcal{V}(\mathbf{A}) = \mathrm{HSP}(\mathbf{A}) \quad \text{and} \quad \mathcal{V}_{fin}(\mathbf{A}) = \mathrm{HSP}_{fin}(\mathbf{A}).$$

We say that an equivalence relation $\sim$ on A is a *congruence* of $\mathbf{A}$ if it is a subalgebra of $\mathbf{A}^2$. Every congruence $\sim$ of $\mathbf{A}$ determines a *quotient* algebra $\mathbf{A}/{\sim}$.

A *term* t in a signature Σ is a formal expression built from variables and symbols in Σ that syntactically describes the composition of basic operations. For an algebra $\mathbf{A}$ over Σ a *term operation* $t^{\mathbf{A}}$ is an operation obtained by composing the basic operations of $\mathbf{A}$ according to t. Let s and t be a pair of terms in a signature Σ. We say that $\mathbf{A}$ satisfies the *identity* $s \approx t$ if the term operations $s^{\mathbf{A}}$ and $t^{\mathbf{A}}$ are equal. We say that a class of algebras $\mathcal{V}$ over Σ satisfies the identity $s \approx t$ if every algebra in $\mathcal{V}$ does.

It follows from Birkhoff's theorem [2] that the variety $\mathcal{V}(\mathbf{A})$ is the class of algebras that satisfy all the identities satisfied by $\mathbf{A}$. An algebra $\mathbf{A}$ is *finitely generated* if there exists a finite subset F of its domain such that the only subalgebra of $\mathbf{A}$ containing F is $\mathbf{A}$. If $\mathbf{A}$ is finite then $\mathcal{V}(\mathbf{A})$ is *locally finite*, i.e., every finitely generated algebra in $\mathcal{V}(\mathbf{A})$ is finite.

3 Core Valued Constraint Languages

For each valued constraint language Γ there is an associated algebra. It has universe D and the set of operations $\mathrm{Pol}(\Gamma)$. If all operations of any given algebra

satisfy the identity $f(x,\dots,x) \approx x$ (i.e. are *idempotent*) then we call the algebra *idempotent*. In this section we prove that every valued constraint language which is finite has a computationally equivalent valued constraint language whose associated algebra is idempotent.

Positive Clone. Those polymorphisms of a given language Γ which are assigned a positive weight by some weighted polymorphisms $\omega \in \mathrm{wPol}(\Gamma)$ are of special interest in the rest of the paper. We begin this section by proving that they form a clone.

Let $\mathcal{C}$ be a weighted clone over a set D. The following proposition shows that the set $\bigcup_{\omega\in\mathcal{C}} \mathrm{supp}(\omega)$, together with the set of projections Π_D, is a clone. We call it the *positive clone* of $\mathcal{C}$ and denote by C^+ (if $\mathcal{C}$ is $\mathrm{wPol}(\Gamma)$ then C^+ is denoted by $\mathrm{Pol}^+(\Gamma)$). Details can be found in Appendix A.1.

Proposition 7. *If $\mathcal{C}$ is a weighted clone then C^+ is a clone.*

Cores. Let Γ be a valued constraint language with a domain D. For $S \subseteq D$ we denote by $\Gamma[S]$ the valued constraint language defined on a domain S and containing the restriction of every cost function $\varrho \in \Gamma$ to S.

By generalizing the arguments for finite-valued languages given in [13, 22], we show (see Appendix A.2) that Γ has a computationally equivalent valued constraint language Γ' such that $\mathrm{Pol}^+_1(\Gamma')$ contains only bijective operations. Such a language is called a *core*. Moreover, Γ' can be chosen to be equal to $\Gamma[S]$ for some $S \subseteq D$.

Proposition 8. *For every valued constraint language Γ there exists a core language Γ', such that the valued constraint language Γ is tractable if and only if Γ' is tractable, and it is NP-hard if and only if Γ' is NP-hard.*

For core languages we characterize the set of unary weighted polymorphisms as consisting of all weightings that assign positive weights only to bijective operations preserving all cost functions (see Appendix A.3).

The proposition below witnesses the importance of the positive clone and is used to prove further results in the subsequent sections. Let Γ be a valued constraint language over a domain D which is finite and a core. For each arity m we fix an enumeration of all the elements of D^m. This allows us to treat every m-ary operation $f \in \mathcal{O}_D^{(m)}$ as a $|D^m|$-tuple. We define a $|D^m|$-ary cost function in $\mathrm{wRelClo}(\Gamma)$ that precisely distinguishes the m-ary operations in the positive clone from all the other m-ary polymorphisms. Details are in Appendix A.4.

Proposition 9. *Let Γ be a valued constraint language over a domain D which is finite and a core. For every m there exists a cost function $\varrho\colon \mathcal{O}_D^{(m)} \to \overline{Q}$ in $\mathrm{wRelClo}(\Gamma)$, and a rational number P, such that for every $f \in \mathcal{O}_D^{(m)}$ the following conditions are satisfied:*

1. *$\varrho(f) \geq P$,*
2. *$\varrho(f) < \infty$ if and only if $f \in \mathrm{Pol}(\Gamma)$,*
3. *$\varrho(f) = P$ if and only if $f \in \mathrm{Pol}^+(\Gamma)$.*

Rigid cores. We further reduce the class of languages that we need to consider. Let Γ be a valued constraint language over an n-element domain $D = \{d_1, \ldots, d_n\}$ which is finite and a core. For each $i \in \{1, \ldots, n\}$, let

$$N_i(x) = \begin{cases} 0 & \text{if } x = d_i, \\ \infty & \text{otherwise,} \end{cases}$$

and let Γ_c denote the valued constraint language obtained from Γ by adding all cost functions N_i. Observe that $\mathrm{Pol}(\Gamma_c) = \mathrm{IdPol}(\Gamma)$, where by $\mathrm{IdPol}(\Gamma)$ we denote the set of idempotent polymorphisms of the language Γ. Hence, the only unary polymorphism of Γ_c is the identity, which also means that there is only one unary weighted polymorphism of Γ_c – the zero-valued weighted polymorphism.

A valued constraint language Γ is a *rigid core* if there is exactly one unary polymorphism of Γ, which is the identity. This notion corresponds to the classical notion of a rigid core considered in CSP [7]. The following proposition, together with Proposition 8, implies that for each finite language Γ, there is a computationally equivalent language that is a rigid core. For details see Appendix A.5.

Proposition 10. *Let Γ be a valued constraint language which is finite and a core. The valued constraint language Γ_c is a rigid core. Moreover, Γ is tractable if and only if Γ_c is tractable, and Γ is NP-hard if and only if Γ_c is NP-hard.*

If Γ is a core language then the positive clone of Γ_c contains precisely the idempotent operations from the positive clone of Γ (see Appendix A.6).

4 Weighted varieties

One of the fundamental results of the algebraic approach to CSP [3, 7, 18] says that the complexity of a crisp language Γ depends only on the variety generated by the algebra $(D, \mathrm{Pol}(\Gamma))$. We generalize this fact to VCSP.

A k-ary *weighting* ω of an algebra $\mathbf{A}$ is a function that assigns rational weights to all k-ary term operations of $\mathbf{A}$ in such a way, that the sum of all weights is 0, and if $\omega(f) < 0$ then f is a projection. A (*proper*) *superposition* $\omega[g_1, \ldots, g_k]$ of a weighting ω with a list of l-ary term operations $g_1, \ldots, g_k$ from $\mathbf{A}$ is defined the same way as for clones (see Definition 5). An algebra $\mathbf{A}$ together with a set of weightings closed under non-negative scaling, addition of weightings of equal arity and proper superposition with term operations from $\mathbf{A}$ is called a *weighted algebra*.

For a variety $\mathcal{V}$ over a signature Σ and a term t we denote by $[t]_{\mathcal{V}}$ the equivalence class of t under the relation $\approx_{\mathcal{V}}$ such that $t \approx_{\mathcal{V}} s$ if and only if the variety $\mathcal{V}$ satisfies the identity $t \approx s$ (we skip the subscript, writing $[t]$ instead of $[t]_{\mathcal{V}}$, whenever the variety is clear from the context). Observe that if the variety is locally finite then there are finitely many equivalence classes of terms of a fixed arity [8].

Definition 11. *Let $\mathcal{V}$ be a locally finite variety over a signature Σ. A k-ary* weighting ω *of $\mathcal{V}$ is a function that assigns rational weights to all equivalence classes of k-ary terms over Σ in such a way, that the sum of all weights is 0, and if $\omega([t]) < 0$ then $\mathcal{V}$ satisfies the identity $t(x_1, \dots, x_k) \approx x_i$ for some $i \in \{1, \dots, k\}$. The variety $\mathcal{V}$ together with a nonempty set of weightings is called a* weighted variety.

Take any finite algebra $\mathbf{B} \in \mathcal{V}$. A k-ary weighting ω of $\mathcal{V}$ *induces* a weighting $\omega^{\mathbf{B}}$ of $\mathbf{B}$ in a natural way:

$$\omega^{\mathbf{B}}(f) = \sum_{\{[t] \mid t^{\mathbf{B}} = f\}} \omega([t]).$$

If $\omega([t]) < 0$ then the term operation $t^{\mathbf{B}}$ is a projection, and hence the weighting $\omega^{\mathbf{B}}$ is proper. For a weighted variety $\mathcal{V}$, by $\mathbf{B} \in \mathcal{V}$ we mean the algebra $\mathbf{B}$ together with the set of weightings induced by $\mathcal{V}$.

For every weighting ω of a finite weighted algebra $\mathbf{A}$ there is a corresponding weighting ω of the variety $\mathcal{V}(\mathbf{A})$ defined by $\omega([t]) = \omega(t^{\mathbf{A}})$. It follows from Birkhoff's theorem that it is well defined. A weighted variety $\mathcal{V}(\mathbf{A})$ *generated* by a weighted algebra $\mathbf{A}$ is the variety $\mathcal{V}(\mathbf{A})$ together with the set of weightings corresponding to the weightings of $\mathbf{A}$.

We prove that every finite algebra $\mathbf{B} \in \mathcal{V}(\mathbf{A})$ together with the set of weightings induced by $\mathcal{V}(\mathbf{A})$ is a weighted algebra. The only non-trivial part is to show that $\mathbf{B}$ is closed under proper superpositions (cf. Appendix B.1).

Proposition 12. *For a finite weighted algebra $\boldsymbol{A}$ over a fixed signature Σ and a finite algebra $\boldsymbol{B} \in \mathcal{V}(\boldsymbol{A})$ let $\omega^{\boldsymbol{B}}$ be a k-ary weighting of $\boldsymbol{B}$ induced by the weighted variety $\mathcal{V}(\boldsymbol{A})$. If for some list $f_1^{\boldsymbol{B}}, \dots, f_k^{\boldsymbol{B}}$ of l-ary term operations from $\boldsymbol{B}$ the composition $\omega^{\boldsymbol{B}}[f_1^{\boldsymbol{B}}, \dots, f_k^{\boldsymbol{B}}]$ is proper then it is induced by some valid weighting of $\mathcal{V}(\boldsymbol{A})$.*

For a finite weighted algebra $\mathbf{A}$ let $\mathrm{Imp}(\mathbf{A})$ denote the set of those cost functions on A that are improved by all weightings of $\mathbf{A}$. We prove that for each finite weighted algebra $\mathbf{B} \in \mathcal{V}(\mathbf{A})$ the valued constraint language $\mathrm{Imp}(\mathbf{B})$ is not harder then $\mathrm{Imp}(\mathbf{A})$ i.e.:

Lemma 13. *Let $\boldsymbol{A}$ be a finite weighted algebra and let*

$$\boldsymbol{B} \in P_{fin}(\boldsymbol{A}) \text{ or } \boldsymbol{B} \in S(\boldsymbol{A}) \text{ or } \boldsymbol{B} \in H(\boldsymbol{A}) \text{ or finally } \boldsymbol{B} \in \mathcal{V}(\boldsymbol{A})$$

then a VCSP defined by any finite subset of $\mathrm{Imp}(\boldsymbol{B})$ *reduces in polynomial-time to a VCSP for some finite subset of* $\mathrm{Imp}(\boldsymbol{A})$.

Therefore the complexity of Γ depends only on the weighted variety generated by the weighted algebra $(D, \mathrm{wPol}(\Gamma))$. Details are in Appendix B.2.

5 Dichotomy conjecture

An operation t of arity k is called a *Taylor operation* of an algebra (or a variety), if t is idempotent and for every $j \leq k$ it satisfies an identity of the form

$$t(\square_1, \square_2, \ldots, \square_k) \approx t(\triangle_1, \triangle_2, \ldots, \triangle_k),$$

where all $\square_i$s and $\triangle_i$s are substituted with either x or y, but $\square_j$ is x whenever $\triangle_j$ is y. In this section we prove the following theorem:

Theorem 14. *Let Γ be a finite core valued constraint language. If $\mathrm{Pol}^+(\Gamma)$ does not have a Taylor operation, then Γ is NP-hard.*

We conjecture[3] that these are the only cases of finite core languages which give rise to NP-hard VCSPs.

Conjecture. *Let Γ be a finite core valued constraint language. If $\mathrm{Pol}^+(\Gamma)$ does not have a Taylor operation, then Γ is NP-hard. Otherwise it is tractable.*

For crisp languages $\mathrm{Pol}^+(\Gamma) = \mathrm{Pol}(\Gamma)$. Therefore Theorem 14 generalizes the well-known result of Bulatov, Jeavons and Krokhin [3, 7] concerning crisp core languages. Similarly the above conjecture is a generalization of The Algebraic Dichotomy Conjecture for CSP. Later on we show that it is supported by all known partial results on the complexity of VCSPs.

To prove Theorem 14 we use Proposition 9 and argue that any relation compatible with $\mathrm{Pol}^+(\Gamma)$ can be found as a set of tuples with minimal costs for some cost function improved by $\mathrm{wPol}(\Gamma)$. It is easy to notice that if $\mathrm{Pol}^+(\Gamma)$ does not have a Taylor operation, then such a relation with NP-complete CSP can be constructed. Details can be found in Appendix C.1.

As the Taylor operation is difficult to work with, in the reminder of the section we use a characterization of Taylor algebras as the algebras possessing a cyclic term. If Γ is a finite core constraint language then $(D, \mathrm{IdPol}^+(\Gamma))$ is a finite idempotent algebra. It follows that $\mathrm{IdPol}^+(\Gamma)$, and hence also $\mathrm{Pol}^+(\Gamma)$, has a Taylor operation if and only if it has an idempotent cyclic operation [1].

5.1 Two-element domain

A complete complexity classification for valued constraint languages over a two-element domain was established in [10]. All tractable languages have been defined via multimorphisms, which are a more restricted form of weighted polymorphisms. A k-ary *multimorphism* of a language Γ, specified as a k-tuple $\langle f_1, \ldots, f_k \rangle$ of k-ary operations on D, is a k-ary weighted polymorphism ω of Γ such that for each $i \in \{1, \ldots, k\}$, we have that $\omega(\pi_i) = -\frac{1}{k}$, and $\omega(f_i) = \frac{l}{k}$, where l is the number of times the operation f_i appears in the tuple.

[3] The conjecture was suggested in a conversation by Libor Barto, however it might have appeared independently earlier.

An operation $f \in \mathcal{O}_D^{(3)}$ is called a *majority* operation if for every $x, y \in D$ we have that $f(x,x,y) = f(x,y,x) = f(y,x,x) = x$. Similarly, an operation $f \in \mathcal{O}_D^{(3)}$ is called a *minority* operation if for every $x, y \in D$ it satisfies $f(x,x,y) = f(x,y,x) = f(y,x,x) = y$. We show the following proposition (see Appendix C.2):

Proposition 15. *Let Γ be a finite core valued constraint language on $D = \{0,1\}$. Then $\mathrm{Pol}^+(\Gamma)$ has an idempotent cyclic operation if and only if Γ admits at least one of the following six multimorphisms: $\langle \min, \min \rangle$, $\langle \max, \max \rangle$, $\langle \min, \max \rangle$, $\langle \mathrm{Mjrty}, \mathrm{Mjrty}, \mathrm{Mjrty} \rangle$, $\langle \mathrm{Mnrty}, \mathrm{Mnrty}, \mathrm{Mnrty} \rangle$, $\langle \mathrm{Mjrty}, \mathrm{Mjrty}, \mathrm{Mnrty} \rangle$.*

The proposition fully agrees with the classification of VCSP languages on two-element domain in [10].

5.2 Finite-valued languages

Theorem 16 (Thapper and Živný [22]). *Let Γ be a finite-valued constraint language which is a core. If Γ admits an idempotent cyclic weighted polymorphism of some arity $m > 1$, then Γ is tractable. Otherwise it is NP-hard.*

To show that our conjecture agrees with the above complexity classification in Appendix C.3 we prove the following result (which holds for general-valued languages):

Proposition 17. *Let Γ be a core valued constraint language. Then Γ admits an idempotent cyclic weighted polymorphism of some arity $m > 1$ if and only if $\mathrm{Pol}^+(\Gamma)$ contains an idempotent cyclic operation of the same arity.*

5.3 Conservative languages

A valued constraint language Γ over a domain D is called *conservative* if it contains all $\{0,1\}$-valued unary cost functions on D. An operation $f \in \mathcal{O}_D^{(k)}$ is *conservative* if for every $x_1, \ldots, x_k \in D$ we have that $f(x_1, \ldots, x_k) \in \{x_1, \ldots, x_k\}$, and a weighted polymorphism is *conservative* if its support contains conservative operations only.

A *Symmetric Tournament Pair* (*STP*) is a conservative binary multimorphism $\langle \sqcap, \sqcup \rangle$, where both operations are commutative, i.e., $\sqcap(x,y) = \sqcap(y,x)$ and $\sqcap(x,y) = \sqcap(y,x)$ for all $x, y \in D$, and moreover $\sqcap(x,y) \neq \sqcup(x,y)$ for all $x \neq y$. A *MJN* is a ternary conservative multimorphism $\langle \mathrm{Mj}_1, \mathrm{Mj}_2, \mathrm{Mn}_3 \rangle$, such that $\mathrm{Mj}_1, \mathrm{Mj}_2$ are majority operations, and Mn_3 is a minority operation.

Theorem 18 (Kolmogorov and Živný [17]). *Let Γ be a conservative constraint language over a domain D. If Γ admits a conservative binary multimorphism $\langle \sqcap, \sqcup \rangle$ and a conservative ternary multimorphism $\langle \mathrm{Mj}_1, \mathrm{Mj}_2, \mathrm{Mn}_3 \rangle$, and there is a family M of two-element subsets of D, such that:*

- *for every $\{x, y\} \in M$, $\langle \sqcap, \sqcup \rangle$ restricted to $\{x, y\}$ is an STP,*

– *for every* $\{x, y\} \notin M$, $\langle \mathrm{Mj}_1, \mathrm{Mj}_2, \mathrm{Mn}_3 \rangle$ *restricted to* $\{x, y\}$ *is an MJN,*

then Γ *is tractable. Otherwise it is NP-hard.*

In this case, as well as in the others, it can be shown (cf. Appendix C.4) that the existence of an idempotent cyclic polymorphism in $\mathrm{Pol}^+(\Gamma)$ is equivalent (for conservative Γ) to the tractability conditions from the theorem above.

Acknowledgments

We are grateful to Libor Barto and Jakub Bulin for inspiring discussions on VCSP.

References

1. Libor Barto and Marcin Kozik. Absorbing subalgebras, cyclic terms, and the constraint satisfaction problem. *Logical Methods in Computer Science*, 8(1), 2012.
2. G. Birkhoff. On the structure of abstract algebras. In *Proceedings of the Cambridge Philosophical Society*, volume 31, pages 433–454, 1935.
3. Andrei Bulatov, Peter Jeavons, and Andrei Krokhin. Classifying the complexity of constraints using finite algebras. *SIAM Journal on Computing*, 34:720–742, 2005.
4. Andrei A. Bulatov. Tractable conservative constraint satisfaction problems. In *Proc. of the 18th Symposium on Logic in Computer Science*, page 321, 2003.
5. Andrei A. Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *J. ACM*, 53(1):66–120, 2006.
6. Andrei A. Bulatov. Complexity of conservative constraint satisfaction problems. *ACM Trans. Comput. Logic*, 12(4):24:1–24:66, 2011.
7. Andrei A. Bulatov, Andrei A. Krokhin, and Peter Jeavons. Constraint satisfaction problems and finite algebras. In *Proc. ICALP '00*, pages 272–282, 2000.
8. S. Burris and H.P. Sankappanavar. *A course in universal algebra.* Graduate texts in mathematics. Springer-Verlag, 1981.
9. David A. Cohen, Martin C. Cooper, Páidí Creed, Peter G. Jeavons, and Stanislav Živný. An algebraic theory of complexity for discrete optimization. *SIAM J. Comput.*, 42(5):1915–1939, 2013.
10. David A. Cohen, Martin C. Cooper, Peter G. Jeavons, and Andrei A. Krokhin. The complexity of soft constraint satisfaction. *Artif. Intell.*, 170(11):983–1016, 2006.
11. Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory. *SIAM J. Comput.*, 28(1):57–104, 1999.
12. Pavol Hell and Jaroslav Nešetřil. On the complexity of h-coloring. *Journal of Combinatorial Theory, Series B*, 48(1):92 – 110, 1990.
13. Anna Huber, Andrei Krokhin, and Robert Powell. Skew bisubmodularity and valued CSPs. In *Proc. SODA '13*, pages 1296–1305. SIAM, 2013.
14. Peter Jeavons, David Cohen, and Marc Gyssens. Closure properties of constraints. *J. ACM*, 44(4):527–548, 1997.
15. Peter Jeavons, Andrei Krokhin, and Stanislav Živný. The complexity of valued constraint satisfaction. *Bulletin of the EATCS*, 113:21–55, 2014.
16. Vladimir Kolmogorov, Johan Thapper, and Stanislav Živný. The power of linear programming for general-valued CSPs. *CoRR*, abs/1311.4219, 2013.

17. Vladimir Kolmogorov and Stanislav Živný. The complexity of conservative valued CSPs. *J. ACM*, 60(2):10:1–10:38, 2013.
18. Benoit Larose and Pascal Tesson. Universal algebra and hardness results for constraint satisfaction problems. *Theor. Comput. Sci.*, 410:1629 – 1647, 2009.
19. Thomas J. Schaefer. The complexity of satisfiability problems. In *Proc. of the 10th ACM Symp. on Theory of Computing*, STOC '78, pages 216–226, 1978.
20. Alfred Tarski. A remark on functionally free algebras. *Annals of Mathematics*, 47(1):163–166, 1946.
21. Walter Taylor. Varieties obeying homotopy laws. *Canad. J. Math.*, 29(3):498527, 1997.
22. Johan Thapper and Stanislav Živný. The complexity of finite-valued CSPs. In *Proc. of the 45th ACM Symp. on Theory of Computing*, STOC '13, pages 695–704, 2013.

A Proofs for Section 3

A.1 Proof of Proposition 7

We will use the following technical lemma (Lemma 6.5 from [9]). It implies that any weighting that can be expressed as a weighted sum of arbitrary superpositions can also be expressed as a superposition of a weighted sum of proper superpositions.

Lemma 19. *Let $\mathcal{C}$ be a weighted clone, and let ω_1 and ω_2 be weightings in $\mathcal{C}$, of arity k and l respectively. For any m-ary operations $f_1, \ldots, f_k, g_1, \ldots, g_l$ of C:*

$$c_1\omega_1[f_1, \ldots, f_k] + c_2\omega_2[g_1, \ldots, g_l] = \omega[f_1, \ldots, f_k, g_1, \ldots, g_l],$$

where

$$\omega = c_1\omega_1[\pi_1^{(k+l)}, \ldots, \pi_k^{(k+l)}] + c_2\omega_2[\pi_{k+1}^{(k+l)}, \ldots, \pi_{k+l}^{(k+l)}].$$

Proof. (of Proposition 7) We need to show that the set C^+ is closed under superposition. Take a k-ary operation f and a list of l-ary operations $g_1, \ldots, g_k$ that all belong to C^+.

If f is a projection there is nothing to prove. Otherwise there is a weighting $\omega \in \mathcal{C}$ such that $\omega(f) > 0$. Similarly for each g_i which is not a projection we find ω_i such that $\omega_i(g_i) > 0$ (if g_i is a projection we put ω_i to be the zero-valued l-ary weighting).

Now, there exist non-negative rational numbers i_j such that the sum

$$\omega[g_1, \ldots, g_k] + i_1\omega_1[\pi_1^l, \ldots, \pi_l^l] + \cdots + i_k\omega_k[\pi_1^l, \ldots, \pi_l^l]$$

is a valid weighting. By Lemma 19 this weighting can be obtained as a superposition of a sum of proper superpositions and therefore belongs to $\mathcal{C}$ which finishes the proof.

A.2 Proof of Proposition 8

We need an auxiliary lemma.

Lemma 20. *For a valued constraint language Γ, let $f \in \mathrm{Pol}_1^+(\Gamma)$ and let $\mathcal{I} \in \mathrm{VCSP}(\Gamma)$. If s is an optimal assignment for $\mathcal{I}$, then $f(s)$ is also optimal.*

Proof. Let f, $\mathcal{I}$ and s be like in the statement of the lemma. Observe that $Cost_{\mathcal{I}}$ (see Definition 1) can be seen as a cost function whose arity is equal to the number of variables in $\mathcal{I}$. Moreover, $Cost_{\mathcal{I}}$ belongs to $\mathrm{wRelClo}(\Gamma)$ as it is clearly expressible over Γ. If $Cost_{\mathcal{I}}(s) = \infty$ then there is no assignment with a finite cost and we are done.

Assume that $Cost_{\mathcal{I}}(s) < \infty$, which means that $s \in \mathrm{Feas}(Cost_{\mathcal{I}})$. If $f \neq \mathrm{id}$, then there exists a weighted polymorphism ω with $\omega(f) > 0$. By definition the following inequality is satisfied:

$$\sum_{g \in \mathrm{Pol}_1(\Gamma)} \omega(g) \cdot Cost_{\mathcal{I}}(g(s)) \leq 0.$$

Without loss of generality we can assume that $\omega(\mathrm{id}) = -1$. Then we have that $\sum_{g \in \mathrm{supp}(\omega)} \omega(g) = 1$ and the inequality above can be rewritten as

$$\sum_{g \in \mathrm{supp}(\omega)} \omega(g) \cdot Cost_{\mathcal{I}}(g(s)) \leq Cost_{\mathcal{I}}(s).$$

On the other hand,

$$\sum_{g \in \mathrm{supp}(\omega)} \omega(g) \cdot Cost_{\mathcal{I}}(g(s)) \geq \sum_{g \in \mathrm{supp}(\omega)} \omega(g) \cdot Cost_{\mathcal{I}}(s) = Cost_{\mathcal{I}}(s).$$

Therefore $Cost_{\mathcal{I}}(g(s)) = Cost_{\mathcal{I}}(s)$ for each operation $g \in \mathrm{supp}(\omega)$. Since $f \in \mathrm{supp}(\omega)$ and s is optimal, $f(s)$ is also optimal.

Proof. (of Proposition 8) Let Γ be a valued constraint language over a domain D. Suppose that there is a unary polymorphism $f \in \mathrm{Pol}^{+}(\Gamma)$ that is not bijective. Let $\Gamma' = \Gamma[f(D)]$, where $f(D) \subsetneq D$ denotes the range of f. There is a natural correspondence between instances of $\mathrm{VCSP}(\Gamma')$ and instances of $\mathrm{VCSP}(\Gamma)$, induced by the correspondence between functions in Γ and their restrictions in Γ'. For any instance $\mathcal{I}'$ of $\mathrm{VCSP}(\Gamma')$ the corresponding instance $\mathcal{I}$ of $\mathrm{VCSP}(\Gamma)$ has the same variables. The cost function ϱ' in each constraint is replaced by any cost function ϱ from Γ, which is equal to ϱ' when restricted to $f(D)$. We show that $\mathrm{Opt}_{\Gamma}(\mathcal{I}) = \mathrm{Opt}_{\Gamma'}(\mathcal{I}')$.

Any assignment for $\mathcal{I}'$ is also an assignment for $\mathcal{I}$, and hence $\mathrm{Opt}_{\Gamma}(\mathcal{I}) \leq \mathrm{Opt}_{\Gamma'}(\mathcal{I}')$. Furthermore, by Lemma 20 for each s that is an optimal assignment for $\mathcal{I}$, we have

$$Cost_{\mathcal{I}}(s) = Cost_{\mathcal{I}}(f(s)) = Cost_{\mathcal{I}'}(f(s)).$$

Therefore, $\mathrm{Opt}_{\Gamma}(\mathcal{I}) \geq \mathrm{Opt}_{\Gamma'}(\mathcal{I}')$.

It follows that $\mathrm{VCSP}(\Gamma)$ is tractable if and only if $\mathrm{VCSP}(\Gamma')$ is tractable, and it is NP-hard if and only if $\mathrm{VCSP}(\Gamma')$ is NP-hard. Moreover, the valued constraint language Γ' is defined over a smaller domain. We replace Γ with Γ' and repeat this procedure, until we obtain a language Γ' that is a core.

A.3 Unary weighted polymorphisms of a core language

Proposition 21. *Let Γ be a core valued constraint language. A unary weighting ω is a weighted polymorphism of Γ if and only if it assigns positive weights only to such bijective operations $f \in \mathrm{Pol}_1(\Gamma)$ that, for all cost functions $\varrho \in \Gamma$, satisfy $\varrho \circ f = \varrho$.*

Proof. If a valid unary weighting ω assigns positive weights only to such operations $f \in \mathrm{Pol}_1(\Gamma)$ that, for all cost functions $\varrho \in \Gamma$, satisfy $\varrho \circ f = \varrho$, then for each $\varrho \in \Gamma$ and a tuple $\mathbf{x} \in \mathrm{Feas}(\varrho)$

$$\sum_{f \in \mathrm{Pol}_1(\Gamma)} \omega(f) \cdot \varrho(f(\mathbf{x})) = \sum_{f \in \mathrm{Pol}_1(\Gamma)} \omega(f) \cdot \varrho(\mathbf{x}) = 0,$$

and ω is clearly a weighted polymorphism of Γ.

For the other direction, let ω be a unary weighted polymorphism of Γ, such that $\mathrm{supp}(\omega) \neq \emptyset$. Without loss of generality assume that $\omega(\mathrm{id}) = -1$. Since Γ is a core language, the operations $g \in \mathrm{supp}(\omega)$ are bijective. For $\varrho \in \Gamma$ and a tuple $\mathbf{x} \in \mathrm{Feas}(\varrho)$ for which ϱ takes the minimal value, we have

$$\sum_{g \in \mathrm{supp}(\omega)} \omega(g) \cdot \varrho(g(\mathbf{x})) + \omega(\mathrm{id}) \cdot \varrho(\mathbf{x}) \leq 0, \text{ hence}$$

$$\varrho(\mathbf{x}) \geq \sum_{g \in \mathrm{supp}(\omega)} \omega(g) \cdot \varrho(g(\mathbf{x})) \geq \sum_{g \in \mathrm{supp}(\omega)} \omega(g) \cdot \varrho(\mathbf{x}) = \varrho(\mathbf{x}).$$

Therefore $\varrho(g(\mathbf{x})) = \varrho(\mathbf{x})$ for each $g \in \mathrm{supp}(\omega)$, which means that the operations in the support preserve the minimal weight.

Note that, since each $g \in \mathrm{supp}(\omega)$ is bijective, it determines a bijection of the set $\mathrm{Feas}(\varrho)$. We have shown that this bijection preserves the set of tuples with minimal weight. It can be similarly shown by induction that it preserves the set of tuples with any other fixed weight. Hence, we have proved that $\varrho \circ g = \varrho$ for all $g \in \mathrm{supp}(\omega)$.

A.4 Proof of Proposition 9

We need the following technical lemma, which is a variant of the well known Farkas' Lemma used in linear programming:

Lemma 22 (Farkas). *Let S and T be finite sets of indices, where T is a disjoint union of two subsets, $T_{\geq}$ and $T_{=}$. For all $i \in S$, and all $j \in T$, let $a_{i,j}$ and b_j be rational numbers. Exactly one of the following holds:*

- *Either there exists a set of non-negative rational numbers $\{z_i \mid i \in S\}$ and a rational number C such that*

$$\text{for each } j \in T_{\geq}, \quad \sum_{i \in S} a_{i,j} z_i \geq b_j + C,$$

$$\text{for each } j \in T_{=}, \quad \sum_{i \in S} a_{i,j} z_i = b_j + C.$$

- *Or else there exists a set of rational numbers $\{y_j \mid j \in T\}$ such that $\sum_{j \in T} y_j = 0$ and*

$$\text{for each } j \in T_{\geq}, \quad y_j \geq 0,$$

$$\text{for each } i \in S, \quad \sum_{j \in T} y_j a_{i,j} \leq 0,$$

$$\text{and } \sum_{j \in T} y_j b_j > 0.$$

The set $\{y_j \mid j \in T\}$ defined in the lemma is called a *certificate of unsolvability.*

Proof. (of Proposition 9) The cost function ϱ is given by a sum of all cost functions in Γ with positive coefficients that we define later on.

Like in the classical CSP, a cost function whose feasibility relation contains exactly those $|D^m|$-tuples which are m-ary polymorphisms of Γ is defined by:

$$\sum_{\substack{\varrho\in\Gamma\\(\mathbf{a_1},\ldots,\mathbf{a_m})\in(\mathrm{Feas}(\varrho))^m}} \varrho(x_{\mathbf{b_1}},\ldots,x_{\mathbf{b_r}}),$$

where $\mathbf{b_i}(j)=\mathbf{a_j}(i)$, and r is the arity of ϱ. For each summand we introduce a variable $z_{\varrho,\mathbf{a_1},\ldots,\mathbf{a_m}}$ and, for each $f\in\mathrm{Pol}^+_m(\Gamma)$ we write:

$$\sum_{\substack{\varrho\in\Gamma\\(\mathbf{a_1},\ldots,\mathbf{a_m})\in(\mathrm{Feas}(\varrho))^m}} z_{\varrho,\mathbf{a_1},\ldots,\mathbf{a_m}}\,\varrho(f(\mathbf{b_1}),\ldots,f(\mathbf{b_r}))=0+C,$$

while for each $f\in\mathrm{Pol}_m(\Gamma)\setminus\mathrm{Pol}^+_m(\Gamma)$:

$$\sum_{\substack{\varrho\in\Gamma\\(\mathbf{a_1},\ldots,\mathbf{a_m})\in(\mathrm{Feas}(\varrho))^m}} z_{\varrho,\mathbf{a_1},\ldots,\mathbf{a_m}}\,\varrho(f(\mathbf{b_1}),\ldots,f(\mathbf{b_r}))\geq 1+C,$$

where $\mathbf{b_i}(j)=\mathbf{a_j}(i)$, and r is the arity of ϱ.

By putting the above equalities and inequalities together we obtain a system of linear inequalities and equations. By Lemma 22 there are two mutually exclusive possibilities. First, there may exist a set of non-negative rational numbers $z_{\varrho,\mathbf{a_1},\ldots,\mathbf{a_m}}$ and a rational number C, such that this system is satisfied. Then the proposition is proved: items 1. and 3. follow trivially from construction. Item 2. follows by definition of the cost function.

Otherwise, there exists a set $\{y_f \mid f\in\mathrm{Pol}_m(\Gamma)\}$ which forms the certificate of unsolvability. Then let us consider a weighting defined by $\omega(f)=y_f$. If ω is a valid weighting, then it is an m-ary weighted polymorphism of Γ. Moreover, ω assigns to all operations in $\mathrm{Pol}_m(\Gamma)\setminus\mathrm{Pol}^+_m(\Gamma)$ non-negative weights that sum up to a positive number. Hence, for some $h\in\mathrm{Pol}_m(\Gamma)\setminus\mathrm{Pol}^+_m(\Gamma)$, we have $\omega(h)>0$, which contradicts $h\notin\mathrm{Pol}^+_m(\Gamma)$. If it happens that $y_g<0$ for some operation $g\in\mathrm{Pol}^+_m(\Gamma)$ that is not a projection, then there exists an m-ary weighted polymorphism of Γ which assigns a positive weight to g. By scaling it and adding to ω (as in the proof of Proposition 7), we obtain the weighted polymorphism needed for the contradiction.

A.5 Proof of Proposition 10

Proof. Let Γ be a finite core valued constraint language over a domain $D=\{d_1,\ldots,d_n\}$. It follows from Proposition 9 that there exist an n-ary cost function $N\in\mathrm{wRelClo}(\Gamma)$, and positive rational numbers $P<Q$, such that the following conditions are satisfied:

- $N(x_1,\ldots,x_n)=P$ if and only if the unary operation g defined by $d_i\mapsto x_i$ belongs to $\mathrm{Pol}^+(\Gamma)$,

– $N(x_1, \ldots, x_n) > Q$ if and only if the unary operation g defined by $d_i \mapsto x_i$ belongs to $\mathrm{Pol}(\Gamma) \setminus \mathrm{Pol}^+(\Gamma)$,
– otherwise $N(x_1, \ldots, x_n) = \infty$.

Assume without loss of generality that $N \in \Gamma$. We show a polynomial-time Turing reduction from $\mathrm{VCSP}(\Gamma_c)$ to $\mathrm{VCSP}(\Gamma)$.

Let $\mathcal{I}_c = (V_c, D, \mathcal{C}_c)$ be an instance of $\mathrm{VCSP}(\Gamma_c)$. The set of variables V in the new instance $\mathcal{I}$ is a disjoint union of V_c and $\{v_1, \ldots, v_n\}$. For every constraint of the form $((v), N_i)$ in $\mathcal{C}_c$ we:

– add a constraint $((v, v_i), \varrho_=)$, where

$$\varrho_=(x, y) = \begin{cases} 0 & \text{if } x = y, \\ \infty & \text{otherwise} \end{cases}$$

  (this cost function is expressible over every valued constraint language, so without loss of generality we can assume that $\varrho_= \in \Gamma$),
– remove the constraint $((v), N_i)$ from $\mathcal{C}_c$.

We obtain a new set of constraints $\mathcal{C}_1$, where all cost functions are already from Γ.

Let C be the sum of weights that all cost functions in all constraints in $\mathcal{C}_1$ assign to all tuples in their feasibility relations. The final set of constraints $\mathcal{C}$ additionally contains m constraints of the form $((v_1, \ldots, v_n), N)$, where m is big enough to ensure that $m \cdot (Q - P) > C$.

There are three possibilities:

– If $\mathrm{Opt}_\Gamma(\mathcal{I}) = \infty$ then no assignment for $\mathcal{I}_c$ has a finite cost. Suppose otherwise and let s_c be an assignment for $\mathcal{I}_c$ with a finite cost. Then s_c gives rise to an assignment s for $\mathcal{I}$ with a finite cost. It coincides with s_c on V_c and for each $i \in \{1, \ldots, n\}$, we set $s(v_i) = d_i$.
– The optimal assignment s for $\mathcal{I}$ satisfies $N(s(v_1, \ldots, v_n)) = P$. Then the tuple $s(v_1, \ldots, v_n)$ determines a unary operation g, defined by $d_i \mapsto s(v_i)$. The operation g, by the definition of the cost function N, belongs to the positive clone $\mathrm{Pol}^+(\Gamma)$. Hence, g^{-1} also belongs to the positive clone. Since Γ is a core, the assignment $g^{-1}(s)$ is optimal for $\mathcal{I}$. Its restriction onto V_c is an optimal assignment for $\mathcal{I}_c$.
– The optimal assignment s for $\mathcal{I}$ satisfies $N(s(v_1, \ldots, v_n)) > Q$. While there are m constraints of the form $((v_1, \ldots, v_n), N)$, we have

$$Cost_{\mathcal{I}}(s) \geq m \cdot Q > m \cdot P + C.$$

  If there was any assignment s_c for $\mathcal{I}_c$ with a finite cost, the corresponding assignment s for $\mathcal{I}$ would satisfy $Cost_{\mathcal{I}}(s) < m \cdot P + C$, which gives a contradiction, and implies that $\mathrm{Opt}_{\Gamma_c}(\mathcal{I}_c) = \infty$.

A.6 Positive clone of a rigid core

Proposition 23. *Let Γ be a valued constraint language which is a core. Then* $\mathrm{IdPol}^+(\Gamma) = \mathrm{Pol}^+(\Gamma_c)$.

We first prove the following lemma:

Lemma 24. *Let Γ be a core valued constraint language. For every weighted polymorphism $\omega \in \mathrm{wPol}(\Gamma)$ there exists an idempotent weighted polymorphism $\omega' \in \mathrm{wPol}(\Gamma)$ such that $\mathrm{supp}(\omega) \cap \mathrm{IdPol}(\Gamma) \subseteq \mathrm{supp}(\omega')$. Moreover, if ω is cyclic then ω' can be chosen to be cyclic.*

Proof. Consider a weighted polymorphism $\omega \in \mathrm{wPol}(\Gamma)$. Take a non-idempotent operation $g \in \mathit{supp}(\omega)$ and let h be a unary operation defined by $h(x) = g(x, \ldots, x)$. Since $\mathrm{Pol}^+(\Gamma)$ is a clone of operations, $h \in \mathrm{Pol}^+(\Gamma)$. Then by Proposition 21 the operation h is bijective and preserves all cost functions in Γ. We modify the weighted polymorphism ω by adding $\omega(g)$ to the weight of the idempotent operation $h^{-1} \circ g$ and then assigning weight 0 to the operation g. It is straightforward to check that the new weighting is a weighted polymorphism of Γ. If g is cyclic then so is $h^{-1} \circ g$. Hence if ω is cyclic then so is the new weighting. We repeat this construction for every non-idempotent operation in $\mathrm{supp}(\omega)$. Finally, we obtain an idempotent weighted polymorphism ω' which satisfies the conditions of the lemma.

Proof. (of Proposition 23) Clearly both sets contain all the projections. Let us take $f \in \mathrm{Pol}^+(\Gamma_c)$ that is not a projection and let ω be a weighted polymorphism of Γ_c such that $f \in \mathrm{supp}(\omega)$. There is a corresponding weighted polymorphism ω' of Γ, which is equal to ω on the idempotent operations and equal 0 otherwise. Then we have $f \in \mathrm{supp}(\omega')$. Since f is idempotent it follows that $f \in \mathrm{IdPol}^+(\Gamma)$.

To prove the reverse inclusion consider $f \in \mathrm{IdPol}^+(\Gamma)$ that is not a projection. Let ω be a weighted polymorphism of Γ such that $f \in \mathrm{supp}(\omega)$. By Lemma 24 there exists an idempotent weighted polymorphism ω' of Γ such that $\mathrm{supp}(\omega) \cap \mathrm{IdPol}(\Gamma) \subseteq \mathrm{supp}(\omega')$. The weighting ω'', defined as a restriction of ω' to the idempotent operations, is a weighted polymorphism of Γ_c with $f \in \mathrm{supp}(\omega'')$. Hence $f \in \mathrm{Pol}^+(\Gamma_c)$.

B Proofs for Section 4

B.1 Proof of Proposition 12

In the proof we use Gordan's Theorem.

Theorem 25 (Gordan). *Let S and T be finite sets of indices. For all $i \in S$, and all $j \in T$, let $a_{i,j}$ be rational numbers. Exactly one of the following holds:*

- *Either there exists a set of non-negative rational numbers $\{z_i \mid i \in S\}$ such that*

$$\begin{aligned} &\text{for some } i \in S, \quad z_i > 0, \\ &\text{for each } j \in T, \quad \sum_{i \in S} a_{i,j} z_i = 0. \end{aligned}$$

– *Or else there exists a set of rational numbers* $\{y_j \mid j \in T\}$ *such that*

$$\text{for each } i \in S, \quad \sum_{j \in T} y_j a_{i,j} > 0.$$

Proof. (of Proposition 12) Let $\omega^{\mathbf{B}}$ and $f_1^{\mathbf{B}}, \ldots, f_k^{\mathbf{B}}$ be as in the statement of the proposition. Assume that the weighting $\omega^{\mathbf{B}}[f_1^{\mathbf{B}}, \ldots, f_k^{\mathbf{B}}]$ is proper.

Notice that the following conditions are equivalent:

– the operation $f_i^{\mathbf{B}}$ is the projection π_j on the j-th coordinate,
– there exists a term t such that $t^{\mathbf{B}} = f_i^{\mathbf{B}}$ and $t^{\mathbf{A}}$ is the projection π_j on the j-th coordinate.

For each $i \in \{1, \ldots, k\}$ consider the set F_i of equivalence classes of terms over Σ defined by

$$F_i = \begin{cases} \{[t] \mid t^{\mathbf{A}} = \pi_j\} & \text{if } f_i^{\mathbf{B}} = \pi_j, \\ \{[t] \mid t^{\mathbf{B}} = f_i^{\mathbf{B}}\} & \text{otherwise} \end{cases}$$

(observe that if $f_i^{\mathbf{B}}$ is a projection then F_i contains a single equivalence class). Take ω to be some k-ary weighting of $\mathbf{A}$ that induces $\omega^{\mathbf{B}}$, and let

$$W = \{\omega[t_1^{\mathbf{A}}, \ldots, t_k^{\mathbf{A}}] \mid [t_i] \in F_i\}.$$

Suppose that for some choice of equivalence classes $[t_i] \in F_i$ the superposition $\omega[t_1^{\mathbf{A}}, \ldots, t_k^{\mathbf{A}}]$ is proper. The weighting $\omega[t_1^{\mathbf{A}}, \ldots, t_k^{\mathbf{A}}]$ of $\mathbf{A}$ induces a weighting of $\mathbf{B}$ which is equal to $\omega^{\mathbf{B}}[f_1^{\mathbf{B}}, \ldots, f_k^{\mathbf{B}}]$, thus in this case the proof is concluded.

This shows that a superposition of $\omega^{\mathbf{B}}$ with any list of projections is always induced by some weighting of $\mathbf{A}$.

Now let us deal with the case when none of the weightings in W is proper. Without loss of generality we can assume that the operations $f_1^{\mathbf{B}}, \ldots, f_k^{\mathbf{B}}$ are pairwise distinct (otherwise we replace $\omega^{\mathbf{B}}$ by its superposition with a suitable list of projections) and hence the sets F_i are disjoint. Let $F = \bigcup F_i$. We remove from F the element of F_i if $f_i^{\mathbf{B}}$ is a projection. The removed elements cannot cause a problem and therefore we assume that for every $[t] \in F$ the operation $t^{\mathbf{A}}$ is not a projection. We apply Gordan's Theorem to the following system of linear equations:

$$\sum_{\nu \in W} \nu(t^{\mathbf{A}}) \cdot z_\nu - z_{[t]} = 0, \text{ for each } [t] \in F.$$

If this system has a non-zero solution in non-negative rational numbers then $z_\nu > 0$ for some $\nu \in W$. Observe that the weighting $\upsilon = \sum_{\nu \in W} \nu \cdot z_\nu$ is proper. Indeed, by the definition of a superposition the only non-projections that could be assigned negative weights by υ are the operations $t^{\mathbf{A}}$ where $[t] \in F$. But each such operation $t^{\mathbf{A}}$ is assigned a non-negative weight $z_{[t]}$. Hence, by Lemma 19 the weighting υ is equal to a proper superposition of some weighting of $\mathbf{A}$ with a list of l-ary term operations of $\mathbf{A}$. Finally, let $p = \sum_{\nu \in W} z_\nu > 0$. The weighting $\frac{1}{p}\upsilon$ of $\mathbf{A}$ induces a weighting of $\mathbf{B}$ which is equal to $\omega^{\mathbf{B}}[f_1^{\mathbf{B}}, \ldots, f_k^{\mathbf{B}}]$.

Otherwise, there exists a set $\{y_{[t]} \mid [t] \in F\}$ of rational numbers, such that

$$\text{for each } \nu \in W, \quad \sum_{[t]\in F} y_{[t]} \cdot \nu(t^{\mathbf{A}}) > 0,$$

and $y_{[t]} < 0$ for each $[t] \in F$. For every $i \in \{1, \dots, k\}$ let us choose $[t_i] \in F_i$ satisfying $y_{[t_i]} = \max\{y_{[t]} \mid [t] \in F_i\}$ (if $f_i^{\mathbf{B}}$ is a projection then we choose $[t_i] \in F_i$ to be the only element of F_i and put $y_{[t_i]} = 0$) and consider the weighting $\upsilon = \omega[t_1^{\mathbf{A}}, \dots, t_k^{\mathbf{A}}]$. Notice that υ may assign negative weights only to operations $t_i^{\mathbf{A}}$. Since

$$\sum_{[t]\in F_1} y_{[t]} \cdot \upsilon(t^{\mathbf{A}}) + \dots + \sum_{[t]\in F_k} y_{[t]} \cdot \upsilon(t^{\mathbf{A}}) > 0,$$

then $\sum_{[t]\in F_i} y_{[t]} \cdot \upsilon(t^{\mathbf{A}}) > 0$ for some $i \in \{1, \dots, k\}$. Hence

$$0 < \sum_{[t]\in F_i} y_{[t]} \cdot \upsilon(t^{\mathbf{A}}) \le \sum_{[t]\in F_i} y_{[t_i]} \cdot \upsilon(t^{\mathbf{A}}) = y_{[t_i]} \cdot \sum_{[t]\in F_i} \upsilon(t^{\mathbf{A}}).$$

It follows that $\sum_{[t]\in F_i} \upsilon(t^{\mathbf{A}}) < 0$, which is a contradiction, since $\sum_{[t]\in F_i} \upsilon(t^{\mathbf{A}})$ is the weight that the proper weighting $\omega^{\mathbf{B}}[f_1^{\mathbf{B}}, \dots, f_k^{\mathbf{B}}]$ assigns to the operation $f_i^{\mathbf{B}}$ (which is not a projection).

B.2 Proof of Lemma 13

The proof consists of a sequence of lemmas.

Lemma 26. *Let $\boldsymbol{A}$ be a finite weighted algebra. For any $\boldsymbol{B} \in P_{fin}(\boldsymbol{A})$, there is a polynomial-time reduction of* VCSP(Imp($\boldsymbol{B}$)) *to* VCSP(Imp($\boldsymbol{A}$)).

Proof. Let A^n be the universe of $\mathbf{B}$ and let Γ be a finite subset of Imp($\mathbf{B}$). Take $\varrho \in \Gamma$ to be an r-ary cost function. There is a natural way of defining a corresponding cost function of arity $n \cdot r$ on the set A. We denote this cost function by ϱ'.

Let ω be a k-ary weighting of the weighted algebra $\mathbf{A}$. The corresponding k-ary weighting $\omega^{\mathbf{B}}$ of $\mathbf{B}$ is a weighted polymorphism of ϱ. Then it is not hard to show that ω is a weighted polymorphism of ϱ', as the basic operations of $\mathbf{B}$ are the operations of $\mathbf{A}$ computed coordinate-wise. Hence, each weighting of $\mathbf{A}$ is a weighted polymorphism of ϱ', which means that $\varrho' \in$ Imp($\mathbf{A}$).

For each $\varrho \in \Gamma$ we have defined a corresponding $\varrho' \in$ Imp($\mathbf{A}$). Let $\Gamma' \subseteq$ Imp($\mathbf{A}$) be the (finite) set of all those cost functions.

Now take an arbitrary instance $\mathcal{I} = (V, A^n, \mathcal{C})$ of VCSP(Γ). Replace the domain A^n by A, and each variable $v_i \in V$ by a set of n variables $\{v_i^1, \dots, v_i^n\}$, obtaining a new set of variables V'. In each constraint $(\sigma, \varrho) \in \mathcal{C}$, where ϱ is an r-ary cost function, replace the r-tuple σ of variables from V by the corresponding nr-tuple of variables from V', and the cost function ϱ by the corresponding cost function ϱ' from Γ'. The new instance $\mathcal{I}' = (V', A, \mathcal{C}')$ is an instance of VCSP(Γ'). It is easy to see that there is a one-to-one correspondence between the optimal assignments for $\mathcal{I}$ and the optimal assignments for $\mathcal{I}'$.

Lemma 27. *Let $\boldsymbol{A}$ be a finite weighted algebra. For any $\boldsymbol{B} \in S(\boldsymbol{A})$, there is a polynomial-time reduction of* VCSP(Imp($\boldsymbol{B}$)) *to* VCSP(Imp($\boldsymbol{A}$)).

Notice that Imp($\mathbf{B}$) $\subseteq$ Imp($\mathbf{A}$), so there is nothing to be proved.

Lemma 28. *Let $\boldsymbol{A}$ be a finite weighted algebra. For any $\boldsymbol{B} \in H(\boldsymbol{A})$, there is a polynomial-time reduction of* VCSP(Imp($\boldsymbol{B}$)) *to* VCSP(Imp($\boldsymbol{A}$)).

Proof. By the isomorphism theorem we can consider $\mathbf{B}$ to be a quotient algebra $\mathbf{A}/\sim$ rather than a homomorphic image of $\mathbf{A}$. Let $A/\sim$ be the universe of $\mathbf{B}$ and let Γ be a finite subset of Imp($\mathbf{B}$). Take $\varrho \in \Gamma$ to be a r-ary cost function. We define a corresponding cost function ϱ' of arity r on the set A by $\varrho'(x_1, \dots, x_r) = \varrho([x_1]_\sim, \dots [x_r]_\sim)$.

Let ω be a k-ary weighting of the weighted algebra $\mathbf{A}$. The corresponding k-ary weighting $\omega^{\mathbf{B}}$ of $\mathbf{B}$ is a weighted polymorphism of ϱ. It is not hard to show that ω is a weighted polymorphism of ϱ'. Hence, each weighting of $\mathbf{A}$ is a weighted polymorphism of ϱ', which means that $\varrho' \in$ Imp($\mathbf{A}$).

For each $\varrho \in \Gamma$ we have defined a corresponding $\varrho' \in$ Imp($\mathbf{A}$). Let $\Gamma' \subseteq$ Imp($\mathbf{A}$) be the (finite) set of all those cost functions.

Now take an arbitrary instance $\mathcal{I} = (V, A/\sim, \mathcal{C})$ of VCSP(Γ). Replace the domain $A/\sim$ by A. In each constraint $(\sigma, \varrho) \in \mathcal{C}$ replace the cost function ϱ by a corresponding cost function ϱ' from Γ'. The new instance $\mathcal{I}' = (V, A, \mathcal{C}')$ is an instance of VCSP(Γ').

If $s' \colon V \to A$ is an optimal assignment for $\mathcal{I}'$, then $s \colon V \to A/\sim$ defined by $s(v) = [s'(v)]_\sim$ is an optimal assignment for $\mathcal{I}$. On the other hand, if $s \colon V \to A/\sim$ is an optimal assignment for $\mathcal{I}$, then any assignment $s' \colon V \to A$, such that for each $v \in V$, we have $s'(v) \in s(v)$, is optimal for $\mathcal{I}'$.

C Proofs for Section 5

C.1 Proof of Theorem 14

To prove Theorem 14 we use the following characterization of algebras possessing a Taylor operation:

Theorem 29 (Taylor [21]). *Let $\boldsymbol{A}$ be a finite idempotent algebra, then the following are equivalent:*

- *$\boldsymbol{A}$ has a Taylor operation,*
- *$\mathcal{V}(\boldsymbol{A})$ (equivalently* HS($\boldsymbol{A}$)*) does not contain a two-element algebra whose every term operation is a projection.*

First let us prove an auxiliary lemma.

Lemma 30. *Let Γ be a finite core valued constraint language over a domain D, and let R be an r-ary relation which is compatible with every polymorphism from* Pol$^+(\Gamma)$*. Then there exists a cost function ϱ_R in* wRelClo(Γ)*, such that for every r-tuple $\mathbf{x}$ the following conditions are satisfied:*

– $\varrho_R(\mathbf{x}) \geq 0$ *and*
– $\varrho_R(\mathbf{x}) = 0$ *if and only if* $\mathbf{x} \in R$.

Proof. Let $R = \{\mathbf{x_1}, \ldots, \mathbf{x_m}\}$ be a relation as in the statement of the lemma. By Proposition 9 there exists a cost function $\varrho' \colon \mathcal{O}_D^{(m)} \to \overline{Q}$ in wRelClo(Γ), and a rational number P, such that for every $f \in \mathcal{O}_D^{(m)}$:

– $\varrho'(f) \geq P$,
– $\varrho'(f) < \infty$ if and only if $f \in \mathrm{Pol}(\Gamma)$,
– $\varrho'(f) = P$ if and only if $f \in \mathrm{Pol}^+(\Gamma)$.

Consider the coordinates $\mathbf{b_1}, \ldots, \mathbf{b_r}$, such that $\mathbf{b_i}(j) = \mathbf{x_j}(i)$. Minimising the cost function ϱ' over all the other coordinates we obtain a cost function ϱ, such that for every r-tuple $\mathbf{x}$ the following conditions are satisfied:

– $\varrho(\mathbf{x}) \geq P$ and
– $\varrho(\mathbf{x}) = P$ if and only if $\mathbf{x} \in R$.

Since wRelClo(Γ) is closed under addition of rational constants, we can add the rational number $-P$ to the cost function ϱ obtaining a cost function ϱ_R which satisfies the conditions given by the lemma.

Proof. (of Theorem 14) Let Γ be a finite core valued constraint language over a domain D, and let Γ_c be a rigid core of Γ. Suppose that $\mathrm{Pol}^+(\Gamma)$ does not have a Taylor operation. By Proposition 23 we have that $\mathrm{IdPol}^+(\Gamma) = \mathrm{Pol}^+(\Gamma_c)$. Therefore, $\mathrm{Pol}^+(\Gamma_c)$ does not have a Taylor operation. Below we prove that VCSP(Γ_c) is NP-hard. This implies, by Proposition 10, that VCSP(Γ) is NP-hard which concludes the proof.

Let $\mathbf{A}$ denote the idempotent algebra $\mathrm{Pol}^+(\Gamma_c)$ over the universe D. By Taylor's theorem HS($\mathbf{A}$) contains a two-element algebra $\mathbf{B}$ whose every term operation is a projection. By the isomorphism theorem we can consider $\mathbf{B}$ to be a quotient algebra rather than a homomorphic image of a subalgebra of $\mathbf{A}$. In other words, there exists a binary relation S compatible with $\mathbf{A}$, which is an equivalence relation on some subuniverse D' of D and has two equivalence classes $[d_0]_S$ and $[d_1]_S$. Moreover, the term operations defined on the set of equivalence classes of S using their arbitrarily chosen representatives are all projections.

Every relation is compatible with a two-element algebra whose every term operation is a projection. Consider the relation $R = \{(1,0,0),(0,1,0),(0,0,1)\}$. It corresponds to the One-in-Three Sat problem, which is NP-complete [19]. We define a ternary relation $S_{1\text{IN}3}$ on D' by:

$$S_{1\text{IN}3} = \{(x_1, x_2, x_3) \,:\, \text{exactly one of } x_1, x_2, x_3 \text{ belongs to } [d_1]_S\}.$$

This relation is compatible with $\mathbf{A}$. Hence, there exists a cost function $\varrho_{S_{1\text{IN}3}}$ in wRelClo(Γ_c) satisfying the conditions given by Lemma 30. It follows from Theorem 4 that the valued constraint language Γ_c is NP-hard if and only if the language $\Gamma_c \cup \{\varrho_{S_{1\text{IN}3}}\}$ is NP-hard.

Now, for every instance of One-in-Three Sat it is easy to construct (in polynomial time) an instance $\mathcal{I}$ of VCSP($\Gamma_c \cup \{\varrho_{S_{1\text{IN}3}}\}$) such that the instance of One-in-Three Sat has a solution if and only if the cost of an optimal assignment for $\mathcal{I}$ is 0 (the instance $\mathcal{I}$ uses only the cost function $\varrho_{S_{1\text{IN}3}}$). This finishes the reduction.

C.2 Proof of Proposition 15

On $D = \{0, 1\}$ there are precisely two constant operations, which we denote by Const_0 and Const_1. By Inv we denote the *inversion* operation defined by $\text{Inv}(0) = 1$ and $\text{Inv}(1) = 0$. To prove Proposition 15 we use the following theorem:

Theorem 31 (Cohen et al. [9]). *Let W be a weighted clone on $D = \{0, 1\}$ that contains a weighting which assigns positive weight to at least one operation that is not a projection. Then W contains one of the following nine weightings:*

1. $\langle \text{Const}_0 \rangle$,
2. $\langle \text{Const}_1 \rangle$,
3. $\langle \text{Inv} \rangle$,
4. $\langle \text{min}, \text{min} \rangle$,
5. $\langle \text{max}, \text{max} \rangle$,
6. $\langle \text{min}, \text{max} \rangle$,
7. $\langle \text{Mjrty}, \text{Mjrty}, \text{Mjrty} \rangle$,
8. $\langle \text{Mnrty}, \text{Mnrty}, \text{Mnrty} \rangle$,
9. $\langle \text{Mjrty}, \text{Mjrty}, \text{Mnrty} \rangle$.

Proof. (of Proposition 15) Each of the operations min, max, Mjrty and Mnrty is idempotent and cyclic. If Γ admits at least one of the six multimorphisms listed in the statement of the proposition then obviously $\text{Pol}^+(\Gamma)$ has an idempotent cyclic operation.

For the other direction, let Γ be a finite core valued constraint language on $D = \{0, 1\}$ such that $\text{Pol}^+(\Gamma)$ has an idempotent cyclic operation. Let Γ_c be a rigid core of Γ. By Proposition 23 we have that $\text{IdPol}^+(\Gamma) = \text{Pol}^+(\Gamma_c)$. Therefore, the weighted clone $\text{wPol}(\Gamma_c)$ contains a weighting which assigns positive weight to at least one operation that is not a projection. Then $\text{wPol}(\Gamma_c)$ contains one of the nine weightings listed in Theorem 31. Since the first three of them are not idempotent, it follows that Γ_c, and hence Γ, admits one of the six remaining multimorphisms, which finishes the proof.

C.3 Proof of Proposition 17

One implication is strightforward: if Γ admits an idempotent cyclic weighted polymorphism of some arity $m > 1$ then $\text{Pol}^+(\Gamma)$ contains an idempotent cyclic operation. To show the other implication we use a technique of constructing weighted polymorphism introduced in [16].

The construction of a new weighted polymorphism of arity m is based on grouping operations in $\mathcal{O}_D^{(m)}$ into so-called *collections* and working with weightings that assign the same weight to every operation in a collection.

Let $\mathbb{G}$ be a fixed set of collections, i.e., subsets of $\mathcal{O}_D^{(m)}$, and let $\mathbb{G}^* \subseteq \mathbb{G}$ be a set of collections satisfying some desired property. An *expansion operator* Exp takes a collection $\mathbf{g} \in \mathbb{G}$ and produces a probability distribution δ over $\mathbb{G}$. We say that Exp is *valid* for a language Γ if, for any $\varrho \in \Gamma$ and any $\mathbf{g} \in \mathbb{G}$, the probability distribution $\delta = \mathrm{Exp}(\mathbf{g})$ satisfies

$$\sum_{\mathbf{h} \in \mathbb{G}} \sum_{h \in \mathbf{h}} \frac{\delta(\mathbf{h})}{|\mathbf{h}|} \varrho(h(\mathbf{x_1}, \ldots, \mathbf{x_m})) \leq \sum_{g \in \mathbf{g}} \frac{1}{|\mathbf{g}|} \varrho(g(\mathbf{x_1}, \ldots, \mathbf{x_m})),$$

for any $\mathbf{x_1}, \ldots, \mathbf{x_m} \in \mathrm{Feas}(\varrho)$. We say that the operator Exp is *non-vanishing* (with respect to the pair $(\mathbb{G}, \mathbb{G}^*)$) if, for any $\mathbf{g} \in \mathbb{G}$, there exists a sequence of collections $\mathbf{g}_0, \mathbf{g}_1, \ldots, \mathbf{g}_r$ with $\mathbf{g}_0 = \mathbf{g}$, such that for each $i \in \{0, \ldots, r-1\}$ the collection $\mathbf{g}_{i+1}$ is assigned a non-zero probability by $\mathrm{Exp}(\mathbf{g}_i)$, and $\mathbf{g}_r \in \mathbb{G}^*$.

Lemma 32 (Expansion Lemma [16]). *Let* Exp *be an expansion operator which is valid for the language Γ and non-vanishing with respect to $(\mathbb{G}, \mathbb{G}^*)$. If Γ admits a weighted polymorphism ω with $\mathrm{supp}(\omega) \subseteq \bigcup \mathbb{G}$, then it also admits a weighted polymorphism ω^* with $\mathrm{supp}(\omega^*) \subseteq \bigcup \mathbb{G}^*$.*

Proof. (of Proposition 17) In order to show the remaining implication assume that $\mathrm{Pol}^+(\Gamma)$ contains an idempotent cyclic operation f of arity $m > 1$. There exists a weighted polymorphism ω of Γ such that $f \in \mathrm{supp}(\omega)$.

We define $\sim$ to be the smalles equivalence relation on $\mathcal{O}_D^{(m)}$ such that $g \sim g'$ if $g(x_1, x_2, \ldots, x_k) = g'(x_2, \ldots, x_k, x_1)$. Observe that if $g \sim g'$ and $g \in \mathrm{Pol}(\Gamma)$ then also $g' \in \mathrm{Pol}(\Gamma)$. Let $\mathbb{G}$ consist of the equivalence classes of the relation $\sim$ restricted to $\mathrm{Pol}(\Gamma)$, and let $\mathbb{G}^* \subseteq \mathbb{G}$ be the set of all one-element equivalence classes, i.e., each $\mathbf{g} \in \mathbb{G}^*$ contains a single cyclic operation.

We now define the expansion operator Exp. Take an arbitrary $\mathbf{g} \in \mathbb{G} \setminus \mathbb{G}^*$ (for $\mathbf{g} \in \mathbb{G}^*$ we produce the probability distribution choosing $\mathbf{g}$ with probability 1) and choose a single operation $g \in \mathbf{g}$. Notice that $\mathbf{g} = \{g_1, g_2, \ldots, g_m\}$, where $g_1 = g$ and $g_i(x_1, \ldots, x_n) = g(x_i, x_{i+1}, \ldots, x_{i-1})$ for $i \in \{2, \ldots, m\}$. Consider a weighting

$$\nu = c \cdot (\omega[g_1, g_2, \ldots, g_m] + \omega[g_2, \ldots, g_m, g_1] + \cdots + \omega[g_m, g_1, \ldots, g_{m-1}]),$$

where c is a suitable positive rational, which we define later on[4].

The weighting ν assigns a positive weight to a cyclic operation $f[g_1, g_2, \ldots, g_m]$. This proves that ν is not zero-valued, and hence in the above definition c can be chosen so that the sum of positive weights in ν equals 1. We say that a weighting ω is *weight-symmetric* if $\omega(g) = \omega(g')$ whenever $g \sim g'$. It is easy to check that

[4] Note that the definition of ν does not depend on the choice of g from $\mathbf{g}$.

ν is weight-symmertic. We define $\mathrm{Exp}(\mathbf{g})$ to be a probability distribution δ on $\mathbb{G}$ such that

$$\delta(\mathbf{h}) = \begin{cases} |\mathbf{h}| \cdot \nu(h) & \text{if } \mathbf{h} \subseteq \mathrm{supp}(\nu), \\ 0 & \text{otherwise,} \end{cases}$$

where h is any of the operations in $\mathbf{h}$. We have already pointed out that ν assigns a positive weight to a cyclic operation $f[g_1, g_2, \ldots, g_m]$. It follows that $\mathrm{Exp}(\mathbf{g})$ assigns non-zero probability to the one-element equivalence class $\{f[g_1, g_2, \ldots, g_m]\}$. Therefore, Exp is non-vanishing.

It remains to show that Exp is valid for Γ. Observe that ν assigns negative weights only to the operations in $\mathbf{g}$. Since it is weight-symmetric, $\nu(g) = -\frac{1}{|\mathbf{g}|}$ for every $g \in \mathbf{g}$. The weighting ν might not be valid but it is not difficult to see that it satisfies the condition characterizing weighted polymorphisms, i.e., for any cost function $\varrho \in \Gamma$, and any list of tuples $\mathbf{x_1}, \ldots, \mathbf{x_m} \in \mathrm{Feas}(\varrho)$, we have

$$\sum_{f \in \mathrm{Pol}_m(\Gamma)} \nu(f) \cdot \varrho(f(\mathbf{x_1}, \ldots, \mathbf{x_m})) \leq 0, \text{ hence}$$

$$\sum_{h \in \mathrm{supp}(\nu)} \nu(h) \cdot \varrho(h(\mathbf{x_1}, \ldots, \mathbf{x_m})) \leq \sum_{g \in \mathbf{g}} \frac{1}{|\mathbf{g}|} \cdot \varrho(g(\mathbf{x_1}, \ldots, \mathbf{x_m})), \text{ but}$$

$$\sum_{h \in \mathrm{supp}(\nu)} \nu(h) \cdot \varrho(h(\mathbf{x_1}, \ldots, \mathbf{x_m})) = \sum_{\mathbf{h} \in \mathbb{G}} \sum_{h \in \mathbf{h}} \frac{\delta(\mathbf{h})}{|\mathbf{h}|} \cdot \varrho(h(\mathbf{x_1}, \ldots, \mathbf{x_m})).$$

This proves that Exp is valid, so by Lemma 32 the language Γ admits a weighted polymorphism ω^* whose support contains only cyclic m-ary operations. Moreover, it follows from the proof of the Expansion Lemma in [16] that ω^* can be constructed so that $f \in \mathrm{supp}(\omega^*)$. By Lemma 24 there exists an idempotent weighted polymorphism ω' of Γ such that $\mathrm{supp}(\omega^*) \cap \mathrm{IdPol}(\Gamma) \subseteq \mathrm{supp}(\omega')$. Its support is non-emply and contains cyclic operations only. This concludes the proof.

C.4 Conservative languages case

Observe that every weighted polymorphism of a conservative language Γ is conservative. Indeed, consider a k-ary weighted polymorphism $\omega \in \mathrm{wPol}(\Gamma)$ and take any $x_1, \ldots, x_k \in D$. Let $\varrho \in \Gamma$ be a unary cost function such that $\varrho(x_i) = 0$ for $i \in \{1, \ldots, k\}$ and $\varrho(x) = 1$ otherwise. Then

$$\sum_{g \in \mathrm{supp}(\omega)} \omega(g) \cdot \varrho(g(x_1, \ldots, x_k)) = \sum_{g \in \mathrm{Pol}_1(\Gamma)} \omega(g) \cdot \varrho(g(x_1, \ldots, x_k)) \leq 0,$$

hence for each $g \in \mathrm{supp}(\omega)$ we have that $\varrho(g(x_1, \ldots, x_k)) = 0$, so $g(x_1, \ldots, x_k) \in \{x_1, \ldots, x_k\}$. This implies that the positive clone of a conservative language is idempotent, and hence every conservative language is a core.

We show that our conjecture agrees with the complexity classification for conservative valued constraint languages:

Proposition 33. *Let Γ be a conservative constraint language over a domain D. Then $\mathrm{Pol}^+(\Gamma)$ has an idempotent cyclic operation if and only if Γ admits a conservative binary multimorphism $\langle \sqcap, \sqcup \rangle$ and a conservative ternary multimorphism $\langle \mathrm{Mj}_1, \mathrm{Mj}_2, \mathrm{Mn}_3 \rangle$, and there is a family M of two-element subsets of D, such that:*

- *for every $\{x, y\} \in M$, $\langle \sqcap, \sqcup \rangle$ restricted to $\{x, y\}$ is an STP,*
- *for every $\{x, y\} \notin M$, $\langle \mathrm{Mj}_1, \mathrm{Mj}_2, \mathrm{Mn}_3 \rangle$ restricted to $\{x, y\}$ is an MJN.*

Proof. Let Γ' be the language Γ together with all $\{0, \infty\}$-valued unary cost functions on D. For every weighted polymorphism $\omega \in \mathrm{wPol}(\Gamma)$ there is a corresponding weighted polymorphism of Γ', which is equal to ω on the conservative operations. Therefore $\mathrm{Pol}^+(\Gamma') = \mathrm{Pol}^+(\Gamma)$, and Γ admits a conservative multimorphism $\langle f_1, \ldots, f_k \rangle$ if and only if Γ' does. Now let $\varrho : D \to \overline{\mathbb{Q}}$ be any general-valued unary cost function. Observe that $\varrho \in \mathrm{wRelClo}(\Gamma')$. It follows that without loss of generality we can assume that Γ contains all general-valued unary cost functions. We do so in the rest of the proof. Observe that every polymorphism of such language is conservative.

Assume that Γ admits the two conservative multimorphisms $\langle \sqcap, \sqcup \rangle$ and $\langle \mathrm{Mj}_1, \mathrm{Mj}_2, \mathrm{Mn}_3 \rangle$ described in the statement of the proposition. There are only four idempotent operations on a two-element domain $\{x, y\}$, namely: max, min, π_1 and π_2 (we assume that $\{x, y\} = \{0, 1\}$). Each of the operations $\sqcap$, $\sqcup$ restricted to any two-element subset $\{x, y\}$ of D must be equal to one of those four. Therefore it is not difficult to prove, using $\{0, 1\}$-valued unary cost functions, that for every two-element subset $\{x, y\}$ of D:

- either $\langle \sqcap, \sqcup \rangle$ restricted to $\{x, y\}$ is an STP,
- or $\sqcap$ restricted to $\{x, y\}$ is equal to π_1 and $\sqcup$ restricted to $\{x, y\}$ is equal to π_2 (possibly the other way round).

Let M' be the set of those two-element subsets $\{x, y\}$ of D, for which $\langle \sqcap, \sqcup \rangle$ restricted to $\{x, y\}$ is an STP. Obviously $M \subseteq M'$.

Let $t(x, y, z) = ((x \sqcap y) \sqcup (x \sqcap z)) \sqcup (z \sqcap y)$. For every $\{x, y\} \in M'$ we have that t restricted to $\{x, y\}$ is the majority operation. For every other two-element subset $\{x, y\}$ of D the operation t restricted to $\{x, y\}$ is equal to π_2 or π_3. Now let us define m to be

$$m(x, y, z) = \mathrm{Mj}_1(t(x, y, z), t(y, z, x), t(z, x, y)).$$

Since the operation Mj_1 is idempotent, for every $\{x, y\} \in M'$ the operation m restricted to $\{x, y\}$ is the majority operation. Moreover, if $\{x, y\}$ does not belong to M' then m restricted to $\{x, y\}$ is equal to Mj_1 (with permuted arguments). But Mj_1 restricted to $\{x, y\}$ is the majority operation. Therefore, m is a majority operation on the whole domain D. If there is a majority operation in the idempotent clone $\mathrm{Pol}^+(\Gamma)$ then there is also an idempotent cyclic operation. This finishes the proof of the right-to-left implication.

The proof of the other implication consists of a sequence of claims and heavily relies on the results of [17].

Assume that $\mathrm{Pol}^+(\Gamma)$ has an idempotent cyclic operation. Let M be a set of all two-element subsets $\{x, y\}$ of D for which there exists no binary cost function $\varrho \in \mathrm{wRelClo}(\Gamma)$ such that

$$(x, y), (y, x) \in \mathrm{Feas}(\varrho), \text{ and } \varrho(x, x) + \varrho(y, y) > \varrho(x, y) + \varrho(y, x).$$

We prove that Γ admits a conservative binary multimorphism $\langle \sqcap, \sqcup \rangle$ and a conservative ternary multimorphism $\langle \mathrm{Mj}_1, \mathrm{Mj}_2, \mathrm{Mn}_3 \rangle$ such that:

- for every $\{x, y\} \in M$, $\langle \sqcap, \sqcup \rangle$ restricted to $\{x, y\}$ is an STP,
- for every $\{x, y\} \notin M$, $\langle \mathrm{Mj}_1, \mathrm{Mj}_2, \mathrm{Mn}_3 \rangle$ restricted to $\{x, y\}$ is an MJN.

Consider the weighted algebra $(D, \mathrm{wPol}(\Gamma))$. Observe that every two-element subset $\{x, y\} \subseteq D$ is a subuniverse of D and let $\mathbf{B}$ be the subalgebra with universe $B = \{x, y\}$.

Claim 1. Every two-element weighted subalgebra $\mathbf{B}$ of $(D, \mathrm{wPol}(\Gamma))$ contains the weighting $\langle \min, \max \rangle$ or $\langle \mathrm{Mjrty}, \mathrm{Mjrty}, \mathrm{Mnrty} \rangle$ (we assume that $B = \{0, 1\}$).

Proof. The weighted algebra $(D, \mathrm{wPol}(\Gamma))$ contains a weighting which assigns a positive weight to an idempotent cyclic operation. Therefore its weighted subalgebra $\mathbf{B}$ contains a weighting which assigns a positive weight to at least one operation that is not a projection, and hence it contains one of the nine weightings listed in Theorem 31. The first three of them are not idempotent. Moreover, for each of the weightings: $\langle \min, \min \rangle$, $\langle \max, \max \rangle$, $\langle \mathrm{Mjrty}, \mathrm{Mjrty}, \mathrm{Mjrty} \rangle$, $\langle \mathrm{Mnrty}, \mathrm{Mnrty}, \mathrm{Mnrty} \rangle$ it is easy to find a $\{0, 1\}$-valued unary cost function that is not improved by it. We conclude that $\mathbf{B}$ contains the weighting $\langle \min, \max \rangle$ or $\langle \mathrm{Mjrty}, \mathrm{Mjrty}, \mathrm{Mnrty} \rangle$, which finishes the proof of Claim 1.

Claim 2. Let $\{x, y\}$ be a two-element subset of D. There exists no binary cost function $\varrho \in \mathrm{wRelClo}(\Gamma)$ such that

$$(x, y), (y, x) \in \mathrm{Feas}(\varrho), \text{ and } \varrho(x, x) + \varrho(y, y) > \varrho(x, y) + \varrho(y, x),$$

and at least one of the pairs (x, x), (y, y) belong to $\mathrm{Feas}(\varrho)$.

Proof. Consider the weighted subalgebra $\mathbf{B}$ of $(D, \mathrm{wPol}(\Gamma)$ with the universe $B = \{x, y\}$. Assume that $\{x, y\} = \{0, 1\}$. By Claim 1 the weighted subalgebra $\mathbf{B}$ contains the weighting $\langle \min, \max \rangle$ or $\langle \mathrm{Mjrty}, \mathrm{Mjrty}, \mathrm{Mnrty} \rangle$.

Suppose that $\mathbf{B}$ contains the weighting $\langle \mathrm{Mjrty}, \mathrm{Mjrty}, \mathrm{Mnrty} \rangle$ and let ω be the weighted polymorphims of Γ that induces $\langle \mathrm{Mjrty}, \mathrm{Mjrty}, \mathrm{Mnrty} \rangle$. Let ϱ be a binary cost function like in the statement of the claim. Without loss of generality assume that $(x, x) \in \mathrm{Feas}(\varrho)$. Then

$$\begin{aligned}
&\sum_{g \in \mathrm{Pol}_3(\Gamma)} \omega(g) \cdot \varrho(g((x, y), (y, x), (x, x))) = \frac{1}{3}\big(2\varrho(\mathrm{Mjrty}((x, y), (y, x), (x, x))) + \\
&+ \varrho(\mathrm{Mnrty}((x, y), (y, x), (x, x))) - \varrho(x, y) - \varrho(y, x) - \varrho(x, x)\big) = \\
&= \frac{1}{3}\big(2\varrho(x, x) + \varrho(y, y) - \varrho(x, y) - \varrho(y, x) - \varrho(x, x)\big) = \\
&= \frac{1}{3}\big(\varrho(x, x) + \varrho(y, y) - \varrho(x, y) - \varrho(y, x)\big) > 0.
\end{aligned}$$

It follows that ϱ is not improved by ω, so $\varrho \notin \mathrm{wRelClo}(\Gamma)$. Similarly we show that if $\mathbf{B}$ contains the weighting $\langle \min, \max \rangle$ and ω is the weighted polymorphims of Γ that induces $\langle \min, \max \rangle$, then ϱ is not improved by ω.

Claim 2 together with Theorem 9 of [17] implies that Γ admits a conservative binary multimorphism $\langle \sqcap, \sqcup \rangle$ such that:

- for every $\{x, y\} \in M$, $\langle \sqcap, \sqcup \rangle$ restricted to $\{x, y\}$ is an STP,
- if $\{x, y\} \notin M$ then $\sqcap$ restricted to $\{x, y\}$ is equal to π_1 and $\sqcup$ restricted to $\{x, y\}$ is equal to π_2.

Claim 3. There exists an operation m in $\mathrm{Pol}^+(\Gamma)$ which is a majority operation.

Proof. First we prove that there exists an operation Mj in $\mathrm{Pol}^+(\Gamma)$ such that for every $\{x, y\} \notin M$ the operation Mj restricted to $\{x, y\}$ is the majority operation. To this end, take any $\{x, y\} \notin M$. By the definition of M there exists a binary cost function $\varrho \in \mathrm{wRelClo}(\Gamma)$ such that

$$(x, y), (y, x) \in \mathrm{Feas}(\varrho), \text{ and } \varrho(x, x) + \varrho(y, y) > \varrho(x, y) + \varrho(y, x).$$

Moreover, by Claim 2 none of the pairs (x, x), (y, y) belongs to $\mathrm{Feas}(\varrho)$. Therefore:

1. there is no operation in $\mathrm{Pol}^+(\Gamma)$ that restricted to $\{x, y\}$ is the max or min operation (such an operation would not even be a polymorphism of ϱ), and
2. it follows from Claim 1 that there exists an operation f in $\mathrm{Pol}^+(\Gamma)$ such that f restricted to $\{x, y\}$ is the majority operation.

By Proposition 3.1 of [6] it follows from the conditions 1 and 2 above that there exists an operation Mj in $\mathrm{Pol}^+(\Gamma)$ such that for every $\{x, y\} \notin M$ the operation Mj restricted to $\{x, y\}$ is the majority operation.

Let $t(x, y, z) = ((x \sqcap y) \sqcup (x \sqcap z)) \sqcup (z \sqcap y)$. For every $\{x, y\} \in M$ we have that t restricted to $\{x, y\}$ is the majority operation. For every other two-element subset $\{x, y\}$ of D the operation t restricted to $\{x, y\}$ is equal to π_3. Now let us define m to be

$$m(x, y, z) = \mathrm{Mj}(t(x, y, z), t(y, z, x), t(z, x, y)).$$

Since Mj is idempotent, for every $\{x, y\} \in M$ the operation m restricted to $\{x, y\}$ is the majority operation. Moreover, if $\{x, y\}$ does not belong to M then m restricted to $\{x, y\}$ is equal to Mj (with permuted arguments). But Mj restricted to $\{x, y\}$ is the majority operation. Therefore, m is a majority operation on the whole domain D.

By [17] it follows from Claims 2 and 3 that Γ admits a conservative ternary multimorphism $\langle \mathrm{Mj}_1, \mathrm{Mj}_2, \mathrm{Mn}_3 \rangle$ such that for every $\{x, y\} \notin M$, $\langle \mathrm{Mj}_1, \mathrm{Mj}_2, \mathrm{Mn}_3 \rangle$ restricted to $\{x, y\}$ is an MJN, which finishes the proof of Proposition 33.