Computing Science

AN ALGEBRAIC THEORY OF COMPLEXITY FOR VALUED CONSTRAINTS: ESTABLISHING A GALOIS CONNECTION

David A. Cohen
Páidí Creed
Peter G. Jeavons
Stanislav Živný

CS-RR-10-16

Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford OX1 3QD

An Algebraic Theory of Complexity for Valued Constraints: Establishing a Galois Connection[*][†]

David A. Cohen[‡]
Páidí Creed[§]
Peter G. Jeavons[¶]
Stanislav Živný[‖]

Abstract

The complexity of any optimisation problem depends critically on the form of the objective function. Valued constraint satisfaction problems are discrete optimisation problems where the function to be minimised is given as a sum of cost functions defined on specified subsets of variables. These cost functions are chosen from some fixed set of available cost functions, known as a valued constraint language. We show in this paper that when the costs are non-negative rational numbers or infinite, then the complexity of a valued constraint problem is determined by certain algebraic properties of this valued constraint language, which we call *weighted polymorphisms*. We define a Galois connection between valued constraint languages and sets of weighted polymorphisms and show how the closed sets of this Galois connection can be characterised. These results provide a new approach in the search for tractable valued constraint languages.

Keywords: Galois connection, valued constraint satisfaction problems, constraint optimisation, weighted polymorphisms, weighted clones, complexity

[*]This research was supported by EPSRC grant EP/F01161X/1. Stanislav Živný is supported by a Junior Research Fellowship at University College, Oxford.

[†]Further copies of this Research Report may be obtained from the Librarian, Oxford University Computing Laboratory, Computing Science, Wolfson Building, Parks Road, Oxford OX1 3QD, England (Telephone: +44-1865-273837, Email: `library@comlab.ox.ac.uk`).

[‡]Department of Computer Science, Royal Holloway, University of London, UK

[§]Department of Computer Science, Royal Holloway, University of London, UK

[¶]Computing Laboratory, University of Oxford, UK

[‖]Computing Laboratory and University College, University of Oxford, UK

1 Introduction

1.1 Background

Classical constraint satisfaction is concerned with the feasibility of satisfying a collection of constraints. The extension of this framework to include optimisation is now also being investigated and a theory of so-called *soft constraints* is being developed.

Several alternative mathematical frameworks for soft constraints have been proposed in the literature, including the very general frameworks of 'semi-ring based constraints' and 'valued constraints' [6]. For simplicity, we shall adopt the valued constraint framework here as it is sufficiently powerful to model a wide range of optimisation problems [21]. In this framework, every tuple of values allowed by a constraint has an associated *cost*, and the goal is to find an assignment with minimal total cost.

The general constraint satisfaction problem (CSP) is NP-hard, and so is unlikely to have a polynomial-time algorithm. However, there has been much success in finding tractable fragments of the CSP by restricting the types of relation allowed in the constraints. A set of allowed relations has been called a *constraint language* [34]. For some constraint languages the associated constraint satisfaction problems with constraints chosen from that language are solvable in polynomial-time, whilst for other constraint languages this class of problems is NP-hard [35, 34, 27]; these are referred to as *tractable languages* and *NP-hard languages*, respectively. Dichotomy theorems, which classify each possible constraint language as either tractable or NP-hard, have been established for constraint languages over 2-element domains [45], 3-element domains [9], for conservative constraint languages [11], maximal constraint languages [13], languages comprising of a single binary relation without sources and sinks [4] (see also [2]), and languages comprising of a single binary relation that is a special triad [3].

The general *valued* constraint satisfaction problem (VCSP) is also NP-hard, but again we can try to identify tractable fragments by restricting the types of allowed *cost functions* that can be used to define the valued constraints. A set of allowed cost functions has been called a *valued constraint language* [21]. Much less is known about the complexity of the optimisation problems associated with different valued constraint languages, although some results have been obtained for certain special cases. In particular, a complete characterisation of complexity has been obtained for valued constraint languages over a 2-element domain with real-valued or infinite costs [21]. This result generalises a number of earlier results for particular optimisation problems such as Max-Sat [22] and Min-Ones [23].

One class of cost functions has been extensively studied: the so-called *submodular* functions. The problem of minimising a submodular objective function occurs in many diverse application areas, including statistical physics [25], the design of electrical networks [41], and operations research [42, 14, 48]. One of the first problems to be recognised as a case of submodular function minimisation was the Max-Flow/Min-Cut problem [24]. Another class of examples arises in pure mathematics: the rank function of a matroid is always a submodular function [28]. After the first two combinatorial polynomial-time algorithms were proposed for submodular function minimisation [46, 32], a series of faster, fully combinatorial and strongly polynomial-time algorithms for this problem has appeared, see [31] for a survey. The complexity of the best of these algorithms is $O(n^6 + n^5 L)$ where n is the number of variables and L is the time required to evaluate the function [43], see also [33]. More practical cubic time algorithms have been developed for many special cases [20, 39, 40], including the Max-Flow/Min-Cut problem [24], the minimisation of a symmetric submodular function [44], the minimisation of a $\{0, 1\}$-valued submodular function over a 2-element domain [23] and the minimisation of any sum of binary submodular functions over an arbitrary finite domain [18]. There exists a class of submodular functions (of arbitrary arity) which can be expressed as cut functions of graphs, and hence minimised in cubic time. However, not all submodular functions can be expressed in this way, and hence not all VCSPs with submodular cost functions are solvable in cubic time by this reduction (see [49] for details).

The results of [23] show that submodularity is essentially the *only* property giving rise to tractable $\{0, 1\}$-valued constraint languages over a 2-element domain, see [21]. Jonsson et al. [36] generalised this result to 3-element domains, and Deineko et al. [26] recently generalised this result to $\{0, 1\}$-valued constraint languages over arbitrary finite domains containing all constants. Kolmogorov and Živný have recently shown that submodularity is essentially the only reason for tractability for conservative finite-valued VCSPs [38]. However, for more general valued constraint languages, very little is known about the possible tractable cases.

In the classical CSP framework it has been shown that the complexity of any constraint language over any finite domain is determined by certain algebraic properties known as *polymorphisms* [35, 34]. This result has reduced the problem of the identification of tractable constraint languages to that of the identification of suitable sets of polymorphisms. In other words, it has been shown to be enough to study just those constraint languages which are characterised by having a given set of polymorphisms. Using the algebraic approach, considerable progress has now been made towards a complete characterisation of the complexity of constraint languages over finite domains of arbitrary size [27, 12, 4, 1, 3, 5].

In the VCSP framework it has been shown that a more general algebraic property known as a *multimorphism* can be used to analyse the complexity of certain valued constraint languages [19, 15, 21]. Multimorphisms have been used to show that there are precisely eight maximal tractable valued constraint languages over a 2-element domain with real-valued or infinite costs, and each of these is characterised by having a particular form of multimorphism [21]. Furthermore, it was shown that many known maximal tractable valued constraint languages over larger finite domains are precisely characterised by a single multimorphism and that key NP-hard examples have (essentially) no multimorphisms [21, 17]. We discuss the notion of multimorphism in more detail in Appendices A and B, where we show that multimorphisms alone are not sufficient to capture the expressive power of valued constraints.

Cohen et al. [16] later generalised the notion of a multimorphism slightly, to that of a *fractional polymorphism*. They showed that fractional polymorphisms, together with the polymorphisms of the underlying feasibility relations, characterise the complexity of any valued constraint language with non-negative rational or infinite costs over any finite domain [16].

1.2 Contributions

In this paper, we extend the results of [16] by introducing a new algebraic construct which we call a *weighted polymorphism*. We are able to show, using the ideas of [16], that the weighted polymorphisms of a valued constraint language are sufficient on their own to determine the complexity of that language. In addition, we are now able to define a Galois connection between valued constraint languages and sets of weighted polymorphisms, and characterise the closed sets on both sides (see Figure 1 below, on page 12; the terms used in this figure, e.g., wPol, Imp, are defined in Section 4).

The Galois connection we establish here can be applied to the search for tractable valued constraint languages in a very similar way to the application of polymorphisms to the search for tractable constraint languages in the classical CSP. First, we need only consider valued constraint languages characterised by weighted polymorphisms. This greatly simplifies the search for a characterisation of all tractable valued constraint languages. Second, any tractable valued constraint language with finite rational or infinite costs must have a non-trivial weighted polymorphism. Hence the results of this paper provide a powerful new set of tools in the search for a polynomial-time/NP-hard dichotomy for finite-domain optimisation problems defined by valued constraints.

The structure of the paper is as follows. In Section 2 we describe the Valued Constraint Satisfaction Problem and define the notion of expressibility. In Section 3 we recall the so-called algebraic approach to the classical Constraint Satisfaction Problem and show how it fits in the VCSP framework as a special case. In Section 4 we consider valued constraint satisfaction problems with non-negative rational or infinite cost values, and show how the complexity of valued constraint languages in this general framework is characterised by weighted polymorphisms. In Section 5 we give a proof of the main new theorem establishing the Galois connection.

2 Valued Constraint Satisfaction Problems

In the valued constraint framework each constraint has an associated function which assigns a *cost* to each possible assignment of values and these costs are chosen from some *valuation structure*, satisfying the following definition.

Definition 2.1. *A* **valuation structure***, Ω, is a totally ordered set, with a minimum and a maximum element (denoted 0 and ∞), together with a commutative, associative binary* **aggregation operator** *(denoted $\oplus$), such that for all $\alpha, \beta, \gamma \in \Omega$, $\alpha \oplus 0 = \alpha$ and $\alpha \oplus \gamma \geq \beta \oplus \gamma$ whenever $\alpha \geq \beta$.*

Definition 2.2. *Let D be a set and Ω a valuation structure. A function $\phi : D^r \to \Omega$ is called a* **cost function** *of arity r on D taking values in Ω.*

Definition 2.3. *An instance of the* **valued constraint satisfaction problem***, (***VCSP***), is a 4-tuple $\mathcal{P} = \langle V, D, C, \Omega \rangle$ where:*

- *V is a finite set of* **variables***.*

- *D is a set of possible* **values** *that may be assigned to those variables.*
- *Ω is a valuation structure representing possible costs.*
- *C is a multi-set of* **constraints***. Each element of C is a pair $c = \langle \sigma, \phi \rangle$ where σ is a tuple of variables called the* **scope** *of c, and ϕ is a $|\sigma|$-ary cost function on D taking values in Ω.*

Definition 2.4. *For any VCSP instance $\mathcal{P} = \langle V, D, C, \Omega \rangle$, an* **assignment** *for $\mathcal{P}$ is a mapping $s : V \to D$. The* **cost** *of an assignment s, denoted $Cost_{\mathcal{P}}(s)$, is given by the aggregation of the costs for the restrictions of s onto each constraint scope, that is,*

$$Cost_{\mathcal{P}}(s) \stackrel{\text{def}}{=} \bigoplus_{\langle\langle v_1, v_2, \ldots, v_m \rangle, \phi \rangle \in C} \phi(s(v_1), s(v_2), \ldots, s(v_m)).$$

A **solution** *to $\mathcal{P}$ is an assignment with minimal cost, and the question is to find a solution.*

Definition 2.5. *A* **valued constraint language** *is any set Γ of cost functions from some fixed set D to some fixed valuation structure Ω. We define* $\mathrm{VCSP}(\Gamma)$ *to be the set of all VCSP instances in which all cost functions belong to Γ.*

The **complexity** of a valued constraint language Γ will be identified with the complexity of the associated $\mathrm{VCSP}(\Gamma)$, as the following definition indicates.

Definition 2.6. *A valued constraint language Γ is called* **tractable** *if, for every finite subset $\Gamma_f \subseteq \Gamma$, there exists an algorithm solving any instance $\mathcal{P} \in \mathrm{VCSP}(\Gamma_f)$ in polynomial time. Conversely, Γ is called* **NP-hard** *if there is some finite subset $\Gamma_f \subseteq \Gamma$ for which $\mathrm{VCSP}(\Gamma_f)$ is NP-hard.*

We now define a closure operator on cost functions, which adds to a set of cost functions all other cost functions which can be *expressed* using that set, in the sense defined below.

Definition 2.7. *For any VCSP instance $\mathcal{P} = \langle V, D, C, \Omega \rangle$, and any list $L = \langle v_1, \ldots, v_r \rangle$ of variables of $\mathcal{P}$, the* **projection** *of $\mathcal{P}$ onto L, denoted $\pi_L(\mathcal{P})$, is the r-ary cost function defined as follows:*

$$\pi_L(\mathcal{P})(x_1, \ldots, x_r) \stackrel{\text{def}}{=} \min_{\{s:V \to D \;|\; \langle s(v_1), \ldots, s(v_r) \rangle = \langle x_1, \ldots, x_r \rangle\}} Cost_{\mathcal{P}}(s)\,.$$

We say that a cost function ϕ is **expressible** *over a constraint language Γ if there exists a VCSP instance $\mathcal{P} \in \mathrm{VCSP}(\Gamma)$ and a list L of variables of $\mathcal{P}$ such that $\pi_L(\mathcal{P}) = \phi$. We define* $\mathrm{Express}(\Gamma)$ *to be the* **expressive power** *of Γ; that is, the set of all cost functions expressible over Γ.*

Note that the list of variables L may contain repeated entries, and we define the minimum over an empty set of costs to be ∞.

Example 2.8. *Let $\mathcal{P}$ be the VCSP instance with a single variable v and no constraints, and let $L = \langle v, v\rangle$. Then, by Definition 2.7,*

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

Hence for any valued constraint language Γ, over any set D, taking values in any valuation structure Ω, $\mathrm{Express}(\Gamma)$ contains this binary cost function, which will be called the **equality** *cost function.*

The next result shows that expressibility preserves tractability.

Theorem 2.9 ([16])**.** *A valued constraint language Γ is tractable if and only if $\mathrm{Express}(\Gamma)$ is tractable; similarly, Γ is NP-hard if and only if $\mathrm{Express}(\Gamma)$ is NP-hard.*

This result shows that, when trying to identify tractable valued constraint languages, it is sufficient to consider only languages of the form $\mathrm{Express}(\Gamma)$. In the following sections, we will show that such languages can be characterised using certain algebraic properties.

3 Classical Constraint Satisfaction

In this Section we shall consider the special case when the valuation structure Ω consists of just two elements, 0 and ∞. In this case, there is only one possible aggregation operation that satisfies the required conditions, which we will refer to as addition and denote by $+$. Note that $0 + 0 = 0$ and $0 + \infty = \infty + 0 = \infty + \infty = \infty$.

Definition 3.1. *We denote by $\mathbf{R}_D$ the set of all cost functions on a set D taking values in $\Omega = \{0, \infty\}$ and by $\mathbf{R}_D^{(k)}$ the k-ary cost functions in $\mathbf{R}_D$.*

With this very simple valuation structure there is a one-to-one correspondence between the set of cost functions $\mathbf{R}_D$ and the set of all finitary *relations* over D. In this correspondence each cost function ϕ in $\mathbf{R}_D$ is associated with the corresponding relation

$$R(\phi) \overset{\text{def}}{=} \{x \in D^r : \phi(x) < \infty\} .$$

Cost functions in $\mathbf{R}_D$ can also be thought of as *predicates*. Subsets of $\mathbf{R}_D$ are sometimes referred to as *crisp* constraint languages [21] and can be used to model the classical constraint satisfaction problem, or CSP, where each assignment is either allowed (cost 0) or disallowed (cost ∞).

The addition of two cost functions $\phi_1, \phi_2 \in \mathbf{R}_D$ corresponds to performing a relational join operation on the associated relations $R(\phi_1)$ and $R(\phi_2)$ [30]. It also corresponds to taking the conjunction of the associated predicates [12]. Moreover, minimising a cost function $\phi \in \mathbf{R}_D$ over one of its arguments corresponds to taking a relational projection of $R(\phi)$ onto its remaining co-ordinates. It also corresponds to existential quantification of the corresponding predicate over that argument.

Definition 3.2. *A set $\Gamma \subseteq \mathbf{R}_D$ is called a* **relational clone** *if it contains the equality cost function and is closed under*

- *rearrangement of arguments;*
- *addition of cost functions;*
- *minimisation over arbitrary arguments.*

For each $\Gamma \subseteq \mathbf{R}_D$ we define $\mathrm{RelClone}(\Gamma)$ *to be the smallest relational clone containing Γ.*

It is a straightforward consequence of Definitions 2.7 and 3.2 that the expressive power of a crisp constraint language is given by the smallest relational clone containing it, as the next result indicates.

Proposition 3.3. *For any $\Gamma \subseteq \mathbf{R}_D$,* $\mathrm{Express}(\Gamma) = \mathrm{RelClone}(\Gamma)$.

This alternative characterisation for the expressive power of a crisp constraint language was first observed in [34], and used to study the complexity of such languages using tools from universal algebra. We will now give a brief summary of this algebraic approach.

For any finite set D, a function $f : D^k \to D$ is called a k-ary *operation* on D.

Definition 3.4. *We denote by $\mathbf{O}_D$ the set of all finitary operations on D and by $\mathbf{O}_D^{(k)}$ the k-ary operations in $\mathbf{O}_D$.*

Definition 3.5. *The k-ary* **projections** *on D are the operations*

$$e_i^{(k)} : D^k \to D\,, \quad (a_1, \ldots, a_k) \mapsto a_i\,.$$

We denote by $\mathbf{J}_D^{(k)}$ the set of all k-ary projections on D.

Definition 3.6. *Let $f \in \mathbf{O}_D^{(k)}$ and $g_1, \ldots, g_k \in \mathbf{O}_D^{(l)}$. The* **superposition** *of f and $g_1, \ldots, g_k$ is the l-ary operation*

$$f[g_1, \ldots, g_k] : D^l \to D\,, \quad (x_1, \ldots, x_l) \mapsto f(g_1(x_1, \ldots, x_l), \ldots, g_k(x_1 \ldots, x_l))\,.$$

Definition 3.7. *A set $F \subseteq \mathbf{O}_D$ is called a* **clone** *of operations if it contains all the projections on D and is closed under superposition.*

For each $F \subseteq \mathbf{O}_D$ we define $\mathrm{Clone}(F)$ *to be the smallest clone containing F. We denote by* $\mathrm{Clone}^{(k)}(F)$ *the k-ary operations in* $\mathrm{Clone}(F)$.

We can extend k-ary operations to operate on tuples in a natural way, as follows. Let $x_1, \ldots, x_k$ be tuples of length r over a set D. We can obtain another element of D^r by applying f to the x_i co-ordinatewise, as follows:

$$f(x_1, \ldots, x_k) \stackrel{\mathrm{def}}{=} \langle f(x_{1,i}, \ldots, x_{k,i}) \mid i = 1 \ldots r \rangle\,.$$

Definition 3.8. *Let ϕ be a cost function of arity r on a set D and let $f \in \mathbf{O}_D^{(k)}$. We say that f is a* **polymorphism** *of ϕ if, for any $x_1, x_2, \ldots, x_k \in D^r$ such that $\phi(x_i) < \infty$ for $i = 1, \ldots, k$, we have*

$$\phi(f(x_1, x_2, \ldots, x_k)) < \infty\,.$$

If f is a polymorphism of ϕ we say ϕ is **invariant** *under f.*

Definition 3.9. *For any valued constraint language Γ over a set D, we denote by $\mathrm{Pol}(\Gamma)$ the set of all operations on D which are polymorphisms of all cost function $\phi \in \Gamma$ and by $\mathrm{Pol}^{(k)}(\Gamma)$ the k-ary operations in $\mathrm{Pol}(\Gamma)$.*

Definition 3.10. *For any $F \subseteq \mathbf{O}_D$, we denote by $\mathrm{Inv}(F)$ the set of all cost functions in $\mathbf{R}_D$ that are invariant under all operations $f \in F$.*

For any set D, the mappings Pol and Inv form a Galois connection between $\mathbf{O}_D$ and $\mathbf{R}_D$. A characterisation of this Galois connection for finite sets D is given by the following two theorems, originally obtained for sets of relations [29, 7].

Theorem 3.11. *For any finite set D, and any $F \subseteq \mathbf{O}_D$, $\mathrm{Pol}(\mathrm{Inv}(F)) = \mathrm{Clone}(F)$.*

Theorem 3.12. *For any finite set D, and any $\Gamma \subseteq \mathbf{R}_D$, $\mathrm{Inv}(\mathrm{Pol}(\Gamma)) = \mathrm{RelClone}(\Gamma)$.*

As with any Galois connection [8], this means that there is a one-to-one correspondence between clones and relational clones. Together with Proposition 3.3, this result shows that the expressive power of any crisp constraint language Γ on a finite set D corresponds to a particular clone of operations on D. Hence, by Theorem 2.9, the search for tractable crisp constraint languages corresponds to a search for suitable clones of operations [12]. This key observation paved the way for applying deep results from universal algebra in the search for tractable constraint languages [13, 11, 10, 9].

4 VCSPs with Rational Costs

In this section we consider the somewhat more general case where the valuation structure contains all positive rational numbers, as well as zero and infinity.

We shall denote by $\mathbb{Q}_+$ the set of all positive rational numbers together with 0. For the remainder of this paper we will consider the case when the valuation structure $\Omega = \mathbb{Q}_+ \cup \{\infty\}$, and the aggregation operation is the standard addition operation on rationals, $+$ (extended so that $a + \infty = \infty$ and $a\infty = \infty$, for all a). This valuation structure will be denoted $\overline{\mathbb{Q}}_+$. It is sufficiently general to encode many standard optimisation problems; see, for example [21].

Definition 4.1. *We denote by $\mathbf{\Phi}_D$ the set of cost functions on D taking values in $\overline{\mathbb{Q}}_+$ and by $\mathbf{\Phi}_D^{(r)}$ the r-ary cost functions in $\mathbf{\Phi}_D$.*

Definition 4.2. *Any cost function $\phi : D^r \to \Omega$ has an associated cost function which takes only the values 0 and ∞, known as its* **feasibility relation**, *denoted $\mathrm{Feas}(\phi)$, which is defined as follows:*

$$\mathrm{Feas}(\phi)(x_1, \ldots, x_r) \stackrel{\mathrm{def}}{=} \begin{cases} 0 & \text{if } \phi(x_1, \ldots, x_r) < \infty \\ \infty & \text{otherwise} \end{cases}\,.$$

We now define a closure operator on cost functions with rational costs, which adds to a set of cost functions all other cost functions which can be obtained from that set by a certain affine transformation.

Definition 4.3. *We say $\phi, \phi' \in \mathbf{\Phi}_D$ are* **cost-equivalent**, *denoted by $\phi \sim \phi'$, if there exist $\alpha, \beta \in \mathbb{Q}$ with $\alpha > 0$ such that*

$$\phi = \alpha\phi' + \beta\,.$$

We denote by $\Gamma_\sim$ the smallest set of cost functions containing Γ which is closed under cost-equivalence.

The next result shows that adding feasibility relations or cost-equivalent cost functions does not increase the complexity of Γ.

Theorem 4.4 ([16])**.** *For any valued constraint language Γ, we have:*

1. $\Gamma \cup \mathrm{Feas}(\Gamma)$ *is tractable if and only if Γ is tractable, and $\Gamma \cup \mathrm{Feas}(\Gamma)$ is NP-hard if and only if Γ is NP-hard.*

2. $\Gamma_\sim$ *is tractable if and only if Γ is tractable, and $\Gamma_\sim$ is NP-hard if and only if Γ is NP-hard.*

We now introduce an algebraic theory for valued constraints based on the notions of *weighted polymorphisms*, *weighted clones* and *weighted relational clones.*

Definition 4.5. *We say a set $\Gamma \subseteq \mathbf{\Phi}_D$ is a* **weighted relational clone** *if it contains the equality cost function and is closed under*

- *cost-equivalence and feasibility;*
- *rearrangement of arguments;*
- *addition of cost functions;*
- *minimisation over arbitrary arguments.*

For each $\Gamma \subseteq \mathbf{\Phi}_D$ we define $\mathrm{wRelClone}(\Gamma)$ to be the smallest weighted relational clone containing Γ.

It is a straightforward consequence of Definitions 2.7 and 4.5 that, for any valued constraint language $\Gamma \subseteq \mathbf{\Phi}$, the set of cost functions that are cost equivalent to the expressive power of Γ, together with all associated feasibility relations, is given by the smallest weighted relational clone containing Γ, as the next result indicates.

Proposition 4.6. *For any $\Gamma \subseteq \mathbf{\Phi}_D$, $\mathrm{Express}(\Gamma \cup \mathrm{Feas}(\Gamma))_\sim = \mathrm{wRelClone}(\Gamma)$.*

Hence, by Theorem 2.9 and Theorem 4.4, the search for tractable valued constraint languages taking values in $\overline{\mathbb{Q}}_+$ corresponds to a search for suitable weighted relational clones. As has been done in the crisp case (Section 3), we will now proceed to establish an alternative characterisation for weighted relational clones which facilitates this search.

Definition 4.7. *We define a k-ary* **weighted operation** *on a set D to be a* partial function $\omega : \mathbf{O}_D^{(k)} \to \mathbb{Q}$ *such that $\omega(f) < 0$ only if f is a projection and*

$$\sum_{f \in \mathbf{dom}(\omega)} \omega(f) = 0 .$$

The **domain** *of ω, denoted* $\mathbf{dom}(\omega)$*, is the subset of $\mathbf{O}_D^{(k)}$ on which ω is defined. We denote by $ar(\omega) = k$ the arity of ω.*

We denote by $\mathbf{W}_D$ the finitary weighted operations on D and by $\mathbf{W}_D^{(k)}$ the k-ary weighted operations on D.

Definition 4.8. *We say two k-ary weighted operations $\omega, \mu \in \mathbf{W}_D^{(k)}$ are* **weight-equivalent** *if $\mathbf{dom}(\omega) = \mathbf{dom}(\mu)$ and there exists some fixed positive rational c, such that $\omega(f) = c\mu(f)$, for all $f \in \mathbf{dom}(\omega)$.*

Definition 4.9. *For any $\omega_1, \omega_2 \in \mathbf{W}_D^{(k)}$, we define the sum of ω_1 and ω_2, denoted $\omega_1 + \omega_2$, to be the k-ary weighted operation ω with $\mathbf{dom}(\omega) = \mathbf{dom}(\omega_1) \cup \mathbf{dom}(\omega_2)$ and*

$$\omega(f) = \begin{cases} \omega_1(f) + \omega_2(f) & f \in \mathbf{dom}(\omega_1) \cap \mathbf{dom}(\omega_2) \\ \omega_1(f) & f \in \mathbf{dom}(\omega_1) \backslash \mathbf{dom}(\omega_2) \\ \omega_2(f) & f \in \mathbf{dom}(\omega_2) \backslash \mathbf{dom}(\omega_1) \end{cases} . \tag{1}$$

Definition 4.10. *For any $\omega \in \mathbf{W}_D^{(k)}$ and any $g_1, g_2, \ldots, g_k \in \mathbf{O}_D^{(l)}$, we define the* **translation** *of ω by $g_1, \ldots, g_k$, denoted $\omega[g_1, \ldots, g_k]$, to be the* partial function $\omega[g_1, \ldots, g_k]$ *from $\mathbf{O}_D^{(l)}$ to $\mathbb{Q}$ defined by*

$$\omega[g_1, \ldots, g_k](f) \stackrel{\text{def}}{=} \sum_{\substack{f' \in \mathbf{dom}(\omega) \\ f = f'[g_1, \ldots, g_k]}} \omega(f') . \tag{2}$$

The domain of $\omega[g_1, \ldots, g_k]$ is the set of l-ary operations $\{f'[g_1, g_2, \ldots, g_k] \mid f' \in \mathbf{dom}(\omega)\}$.

Example 4.11. *Let ω be the 4-ary weighted operation on D given by*

$$\omega(f) = \begin{cases} -1 & \text{if } f \in \mathbf{J}_D^{(4)} \\ +1 & \text{if } f \in \{\max(x_1, x_2), \min(x_1, x_2), \max(x_3, x_4), \min(x_3, x_4)\} \end{cases} ,$$

and let

$$\langle g_1, g_2, g_3, g_4 \rangle = \left\langle e_1^{(3)}, e_2^{(3)}, e_3^{(3)}, \max(x_1, x_2) \right\rangle .$$

Then, by Definition 4.10 we have

$$\omega[g_1, g_2, g_3, g_4](f) = \begin{cases} -1 & \text{if } f \in \mathbf{J}_D^{(3)} \\ +1 & \text{if } f \in \{\max(x_1, x_2, x_3), \min(x_1, x_2), \min(x_3, \max(x_1, x_2))\} \\ 0 & \text{if } f = \max(x_1, x_2) \end{cases} .$$

Note that $\omega[g_1, g_2, g_3, g_4]$ satisfies the conditions of Definition 4.7 and hence is a weighted operation.

Example 4.12. *Let ω be the same as in Example 4.11 but now consider*

$$\langle g_1', g_2', g_3', g_4' \rangle = \left\langle e_1^{(4)}, \max(x_2, x_3), \min(x_2, x_3), e_4^{(4)} \right\rangle .$$

By Definition 4.10 we have

$$\omega[g_1', g_2', g_3', g_4'](f) = \begin{cases} -1 & \text{if } f \in \{e_1^{(4)}, \max(x_2, x_3), \min(x_2, x_3), e_4^{(4)}\} \\ +1 & \text{if } f \in \left\{ \begin{array}{l} \max(x_1, x_2, x_3), \min(x_1, \max(x_2, x_3)), \\ \max(\min(x_2, x_3), x_4), \min(x_2, x_3, x_4) \end{array} \right\} \end{cases} .$$

Note that $\omega[g_1', g_2', g_3', g_4']$ does not satisfy the conditions of Definition 4.7 because, for example, $\omega[g_1', g_2', g_3', g_4'](f) < 0$ when $f = \max(x_2, x_3)$, which is not a projection. Hence $\omega[g_1', g_2', g_3', g_4']$ is not a weighted operation.

Definition 4.13. *If the result of a translation is a weighted operation, then that translation will be called a* **proper** *translation.*

Remark 4.14. *For any $\omega \in \mathbf{W}_D^{(k)}$, if $g_1, \ldots, g_k$ are* projections, *then it can be shown that the function $\omega[g_1, \ldots, g_k]$ satisfies the conditions of Definition 4.7, and hence is a weighted operation. This means that a translation by any list of projections is always a proper translation.*

We are now ready to define *weighted clones.*

Definition 4.15. *Let C be a clone of operations on D. We say a set $W \subseteq \mathbf{W}_D$ is a* **weighted clone** *with* **support** *C if it contains all zero-valued weighted operations whose domains are subsets of C and is closed under weight-equivalence, addition, and proper translation by operations from C.*

For each $W \subseteq \mathbf{W}_D$ we define $\mathrm{wClone}(W)$ *to be the smallest weighted clone containing W.*

Remark 4.16. *The support of* $\mathrm{wClone}(W)$ *is the clone generated by the domains of the elements of W. That is, the support of* $\mathrm{wClone}(W)$ *is given by* $\mathrm{Clone}(\cup_{\omega \in W} \mathbf{dom}(\omega))$.

Example 4.17. *For any clone of operations, C, there exists a unique weighted clone which consists of all weighted operations assigning weight 0 to each subset of C.*

Definition 4.18. *Let $\phi \in \mathbf{\Phi}_D^{(r)}$ and let $\omega \in \mathbf{W}_D^{(k)}$. We say that ω is a* **weighted polymorphism** *of ϕ if, for any $x_1, x_2, \ldots, x_k \in D^r$ such that $\phi(x_i) < \infty$ for $i = 1, \ldots, k$, we have*

$$\sum_{f \in \mathbf{dom}(\omega)} \omega(f)\phi(f(x_1, x_2, \ldots, x_k)) \leq 0 . \qquad (3)$$

If ω is a weighted polymorphism of ϕ we say ϕ is **improved** *by ω.*

Note that, because $\omega(f)\infty = \infty$ for any value $\omega(f)$, then, if inequality (3) holds we must have $\phi(f(x_1, \ldots, x_k)) < \infty$, for all $f \in \mathbf{dom}(\omega)$. In other words, if ϕ is improved by ω, then every element of $\mathbf{dom}(\omega)$ must be a polymorphism of ϕ.

Example 4.19. *Consider the class of submodular set functions [42]. These are precisely the cost functions on $\{0,1\}$ satisfying*

$$\phi(\min(x_1,x_2)) + \phi(\max(x_1,x_2)) - \phi(x) - \phi(y) \leq 0\,.$$

In other words, the set of submodular functions are defined as the set of cost functions on $\{0,1\}$ with the 2-ary weighted polymorphism ω_{SM} defined by

$$\omega_{SM}(f) = \left\{ \begin{array}{ll} -1 & \text{if } f \in \{e_1^{(2)}, e_2^{(2)}\} \\ +1 & \text{if } f \in \{\min(x_1,x_2), \max(x_1,x_2)\} \end{array} \right. .$$

Definition 4.20. *For any $\Gamma \subseteq \mathbf{\Phi}_D$, we denote by* $\mathrm{wPol}(\Gamma)$ *the set of all finitary weighted operations on D which are weighted polymorphisms of all cost function $\phi \in \Gamma$ and by* $\mathrm{wPol}^{(k)}(\Gamma)$ *the k-ary weighted operations in* $\mathrm{wPol}(\Gamma)$.

Definition 4.21. *For any $W \subseteq \mathbf{W}_D$, we denote by* $\mathrm{Imp}(W)$ *the set of all cost functions in $\mathbf{\Phi}_D$ that are improved by all weighted operations $\omega \in W$ and by* $\mathrm{Imp}^{(r)}(W)$ *the r-ary cost functions in* $\mathrm{Imp}(W)$.

It follows immediately from the definition of a Galois connection [8] that, for any set D, the mappings wPol and Imp form a Galois connection between $\mathbf{W}_D$ and $\mathbf{\Phi}_D$. A characterisation of this Galois connection for finite sets D is given by the following two theorems (see Figure 1 for a diagram).

Theorem 4.22. *For any finite set D, and any finite $\Gamma \subseteq \mathbf{\Phi}_D$,*

$$\mathrm{Imp}(\mathrm{wPol}(\Gamma)) = \mathrm{wRelClone}(\Gamma).$$

Theorem 4.23. *For any finite set D, and any finite $W \subseteq \mathbf{W}_D$,*

$$\mathrm{wPol}(\mathrm{Imp}(W)) = \mathrm{wClone}(W).$$

As with any Galois connection [8], this means that there is a one-to-one correspondence between weighted clones and weighted relational clones. Hence, by Proposition 4.6, Theorem 2.9, and Theorem 4.4, the search for tractable valued constraint languages taking values in $\overline{\mathbb{Q}}_+$ corresponds to a search for suitable weighted clones.

5 Proof of Theorems 4.22 and 4.23

A key tool in proving Theorems 4.22 and 4.23 is the Farkas Lemma from Linear Programming [42, 47]. We state below the variant we use in our proofs.

Lemma 5.1 (Farkas 1894)**.** *Let S and T be finite sets of indices, where T is the 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:*

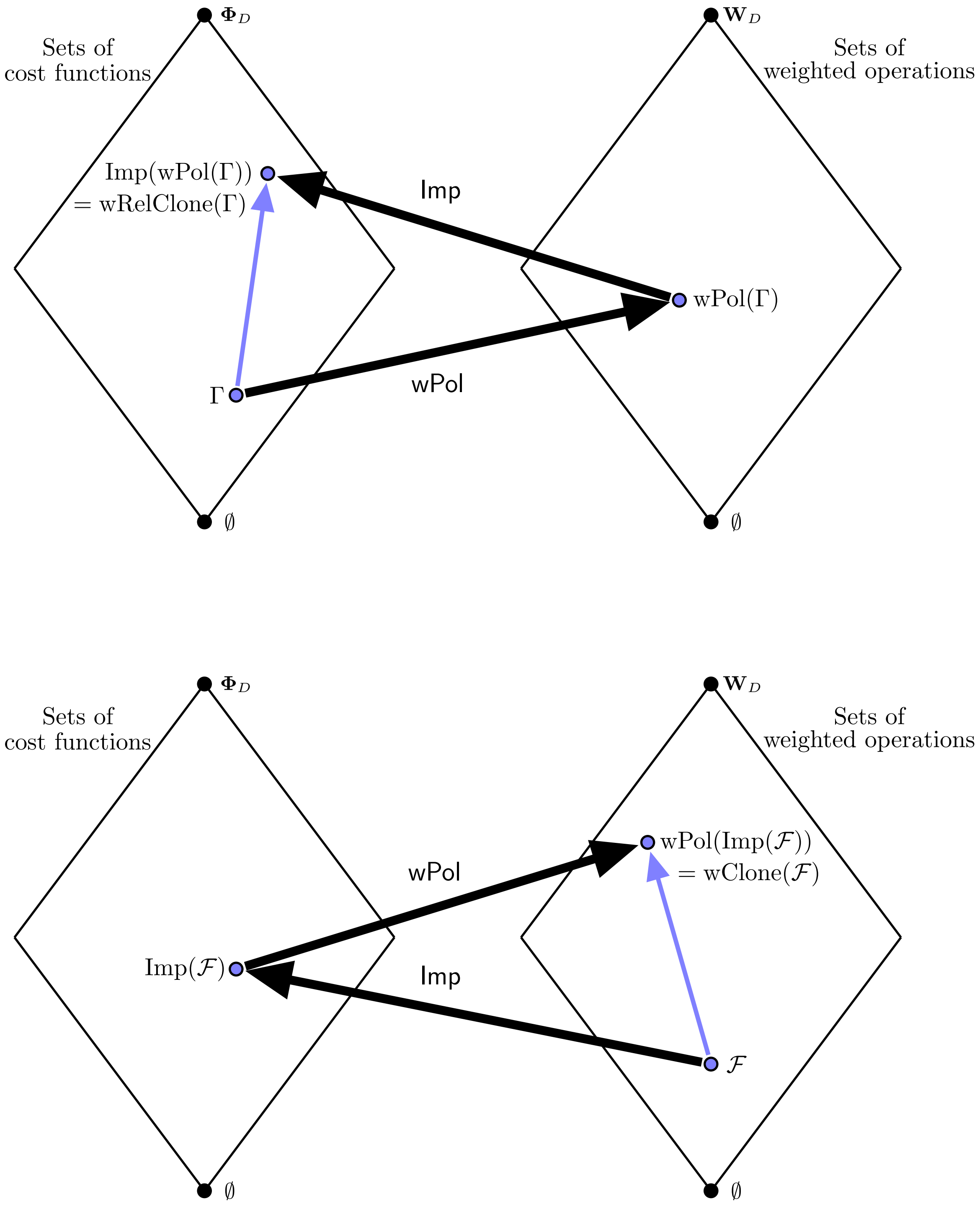
$\mathbf{\Phi}_D$
Sets of cost functions
$\mathbf{W}_D$
Sets of weighted operations
$\mathrm{Imp}(\mathrm{wPol}(\Gamma))$
$= \mathrm{wRelClone}(\Gamma)$
Imp
$\mathrm{wPol}(\Gamma)$
Γ
wPol
$\emptyset$
$\emptyset$
$\mathbf{\Phi}_D$
Sets of cost functions
$\mathbf{W}_D$
Sets of weighted operations
$\mathrm{wPol}(\mathrm{Imp}(\mathcal{F}))$
$= \mathrm{wClone}(\mathcal{F})$
wPol
$\mathrm{Imp}(\mathcal{F})$
Imp
$\mathcal{F}$
$\emptyset$
$\emptyset$

Figure 1: Galois connection between $\mathbf{\Phi}_D$ and $\mathbf{W}_D$.

- *Either there exists a set of non-negative rational numbers* $\{x_i \mid i \in S\}$ *and a rational number* c *such that*

$$\begin{aligned} \textit{for each } j \in T_{\geq}, \quad \sum_{i \in S} a_{i,j}\, x_i \ &\geq\ b_j + c, \qquad \textit{and,} \\ \textit{for each } j \in T_{=}, \quad \sum_{i \in S} a_{i,j}\, x_i \ &=\ b_j + c. \end{aligned}$$

- *Or else there exists a set of integers* $\{y_j \mid j \in T\}$ *such that* $\sum_{j \in T} y_j = 0$ *and:*

$$\begin{aligned} \textit{for each } j \in T_{\geq}, \quad y_j \ &\geq\ 0, \\ \textit{for each } i \in S, \quad \sum_{j \in T} y_j\, a_{i,j} \ &\leq\ 0, \qquad \textit{and} \\ \sum_{j \in T} y_j\, b_j \ &>\ 0. \end{aligned}$$

  Such a set is called a certificate of unsolvability*.*

A similar result to Theorem 4.22 was obtained in [16, Theorem 4] using the related algebraic notion of fractional polymorphism. The proof given in [16] can be adapted in a straightforward way, so we omit the details.

We will prove Theorem 4.23 in two parts. First, we show in Proposition 5.2 that the weighted polymorphisms of a set of cost functions form a weighted clone. Then we show in Theorem 5.4 that any weighted operation that improves all cost functions in $\mathrm{Imp}(W)$ is an element of the weighted clone $\mathrm{wClone}(W)$.

Proposition 5.2. *Let D be a finite set.*

1. *For all* $\Gamma \subset \mathbf{\Phi}_D$*,* $\mathrm{wPol}(\Gamma)$ *is a weighted clone with support* $\mathrm{Pol}(\Gamma)$*.*

2. *For all* $W \subset \mathbf{W}_D$*,* $\mathrm{wClone}(W) \subseteq \mathrm{wPol}(\mathrm{Imp}(W))$*.*

Proof. Certainly $\mathrm{wPol}(\Gamma)$ contains all zero-valued weighted operations with domains contained in $\mathrm{Pol}(\Gamma)$, since all of these satisfy the conditions set out in Definition 4.18. Similarly, $\mathrm{wPol}(\Gamma)$ is closed under addition and weight-equivalence, since both of these operations preserve inequality (3). Hence, to show $\mathrm{wPol}(\Gamma)$ is a weighted clone we only need to show $\mathrm{wPol}(\Gamma)$ is closed under proper translations by members of $\mathrm{Pol}(\Gamma)$.

Let $\omega \in \mathrm{wPol}^{(k)}(\Gamma)$ and suppose $\omega' = \omega[g_1, \ldots, g_k]$ is a proper translation of ω, where $g_1, g_2, \ldots, g_k \in \mathrm{Pol}^{(l)}(\Gamma)$. We will now show that $\omega' \in \mathrm{wPol}^{(l)}(\Gamma)$. Suppose ϕ is an r-ary cost function satisfying $\omega \in \mathrm{wPol}(\{\phi\})$, i.e., ϕ and ω satisfy (3) for any $x_1, x_2, \ldots, x_k \in \mathrm{Feas}(\phi)$. Given any $x'_1, x'_2, \ldots, x'_l \in \mathrm{Feas}(\phi)$, set $x_i = g_i(x'_1, x'_2, \ldots, x'_l)$ for i=1,2,...,k. Then, if we set $f' = f[g_1, \ldots, g_k]$, we have $f'(x'_1, x'_2, \ldots, x'_l) = f(x_1, x_2, \ldots, x_k)$, for any $f \in \mathbf{O}_D^{(k)}$. Hence, by Definition 4.10, we have

$$\sum_{f' \in \mathbf{dom}(\omega')} \omega'(f')\phi(f'(x'_1, x'_2, \ldots, x'_k) = \sum_{f \in \mathbf{dom}(\omega)} \omega(f)\phi(f(x_1, x_2, \ldots, x_k) \leq 0\,.$$

For the second part, we observe that $W \subseteq \mathrm{wPol}(\mathrm{Imp}(W))$ and, therefore, $\mathrm{wClone}(W) \subseteq \mathrm{wClone}(\mathrm{wPol}(\mathrm{Imp}(W))) = \mathrm{wPol}(\mathrm{Imp}(W))$. □

We will make use of the following lemma, which shows that a weighted sum of arbitrary translations of any weighted operations ω_1 and ω_2 can be obtained by translating ω_1 and ω_2 by projection operations, forming a weighted sum, and then translating the result.

Lemma 5.3. *For any weighted operations $\omega_1 \in \mathbf{W}_D^{(k)}$ and $\omega_2 \in \mathbf{W}_D^{(l)}$ and any $g_1, \dots, g_k \in \mathbf{O}_D^{(m)}$ and $g'_1, \dots, g'_l \in \mathbf{O}_D^{(m)}$,*

$$c_1\,\omega_1[g_1, \dots, g_k] + \ c_2\,\omega_2[g'_1, \dots, g'_l] = \omega[g_1, \dots, g_k, g'_1, \dots, g'_l]\,, \tag{4}$$

where $\omega = c_1\,\omega_1[e_1^{(k+l)}, \dots, e_k^{(k+l)}] + \ c_2\,\omega_2[e_{k+1}^{(k+l)}, \dots, e_{k+l}^{(k+l)}]$

Proof. For any $f \in \mathbf{dom}(\omega)$, the result of applying the right-hand side expression in equation (4) to f is:

$$\sum_{\substack{f' \in \mathbf{dom}(\omega) \\ f = f'[g_1,\dots,g_k,g'_1,\dots,g'_l]}} \left(\sum_{\substack{h' \in \mathbf{dom}(\omega_1) \\ f' = h'[e_1^{(k+l)},\dots,e_k^{(k+l)}]}} c_1\,\omega_1(h') \ + \sum_{\substack{h' \in \mathbf{dom}(\omega_2) \\ f' = h'[e_{k+1}^{(k+l)},\dots,e_{k+l}^{(k+l)}]}} c_2\,\omega_2(h') \right).$$

Replacing each f' by the equivalent superposition of h' with projections, we obtain:

$$\sum_{\substack{h' \in \mathbf{dom}(\omega_1) \\ f = h'[g_1,\dots,g_k]}} c_1\,\omega_1(h') + \sum_{\substack{h' \in \mathbf{dom}(\omega_2) \\ f = h'[g'_1,\dots,g'_l]}} c_2\,\omega_2(h')\,,$$

which is the result of applying the left-hand-side of Equation 4 to f. □

Theorem 5.4. *For all finite $W \subset \mathbf{W}_D$,* $\mathrm{wPol}(\mathrm{Imp}(W)) \subseteq \mathrm{wClone}(W)$.

Proof. We prove the theorem as follows. Given a weighted operation $\omega_0 \in \mathbf{W}_D^{(k)}$, we show that either there exists a cost function $\phi \in \mathrm{Imp}(W)$ such that $\omega_0 \notin \mathrm{wPol}(\{\phi\})$ or else ω_0 is equal to a positive weighted sum of translations of weighted operations in W, and hence $\omega_0 \in \mathrm{wClone}(W)$.

Let $M = |D|^k$. We first observe that it is sufficient to consider $\phi \in \mathrm{Imp}^{(M)}(W)$. To see this, suppose there exists a cost function $\phi \in \mathrm{Imp}(W)$ with arity $N > M$ such that $\omega_0 \notin \mathrm{wPol}(\{\phi\})$ and let $x_1, \dots, x_k \in D^N$ be any set of tuples for which the inequality (3) does not hold for ω_0 and ϕ. Let $\mathbf{X}$ be the $k \times N$ matrix whose rows are the tuples $x_1, \dots, x_k$. Since $N > M$ it follows that some of the columns in this matrix must be equal. Moreover, if the i-th and j-th column of $\mathbf{X}$ are equal, then so will be the i-th and j-th entry of the tuple $f(x_1, \dots, x_k)$ obtained by applying any $f \in \mathbf{O}_D^{(k)}$ to these k tuples.

Now let ϕ' be the cost function of arity $\leq M$ that depends only on the first of each of these repeated columns, and takes the same values as ϕ takes on arguments with the appropriate entries repeated. Let $\mathbf{X}'$ be the reduced form of $\mathbf{X}$ (with repeated columns deleted). By this approach, we can construct ϕ' so that $\phi' \in \mathrm{Imp}(W)$, but $\mathbf{X}'$ gives a certificate for $\omega_0 \notin \mathrm{wPol}(\{\phi\})$, i.e., the rows of $\mathbf{X}'$ form a list of tuples for which (3) is violated for ω_0 and ϕ'.

Moreover, if we have a cost function $\phi \in \mathrm{Imp}(W)$ with arity $N < M$ such that $\omega_0 \notin \mathrm{wPol}(\{\phi\})$, then ϕ can be extended to a cost function ϕ' of arity M that does not depend on the $M - N$ added inputs, and, hence, is also contained in $\mathrm{Imp}(W)$ but $\omega_0 \notin \mathrm{wPol}(\{\phi'\})$.

By the argument given above, there exists a cost function $\phi \in \mathrm{Imp}(W)$ such that $\omega_0 \notin \mathrm{wPol}(\{\phi\})$ if and only if there exists a cost function $\phi_M \in \mathrm{Imp}^{(M)}(W)$ such that $\omega_0 \notin \mathrm{wPol}(\{\phi_M\})$. Furthermore, by reordering the arguments of ϕ_M if necessary, we can assume that ϕ_M and ω_0 violate (3) on the particular list of tuples $x_1, \ldots, x_k$ given by taking the rows of a matrix, $\mathbf{X}_M$, whose columns are precisely the tuples in D^k, ordered lexicographically.

By Definition 4.18, such a cost function ϕ_M exists if and only if the following system of inequalities can be satisfied, for all $\omega \in W$ and all $t_1, \ldots, t_{\mathrm{ar}(\omega)} \in D^M$ such that $\phi_M(t_i) < \infty$ for $i = 1, \ldots, \mathrm{ar}(\omega)$,

$$\sum_{g \in \mathbf{dom}(\omega)} \omega(g)\, \phi_M(g(t_1, \ldots, t_{\mathrm{ar}(\omega)})) \;\leq\; 0\,, \tag{5}$$

and, for the tuples $x_1, \ldots, x_k$ forming the rows of $\mathbf{X_M}$, $\phi_M(x_i) < \infty$ for $i = 1, \ldots, k$ and

$$\sum_{f \in \mathbf{dom}(\omega_0)} \omega_0(f)\, \phi_M(f(x_1, \ldots, x_k)) \;>\; 0\,. \tag{6}$$

Every tuple $t \in D^M$ can be viewed as the list of values for a function $t : D^k \to D$. Hence, to satisfy the above system of inequalities, we need to find values in $\overline{\mathbb{Q}}_+$ for the set of variables $\{\phi_M(t) \mid t : D^k \to D\}$.

We now observe that, by inequality (5), for any $\omega \in W$, if $\phi_M(t_i) < \infty$ for $i = 1, \ldots, \mathrm{ar}(\omega)$, then $\phi_M(g(t_1, \ldots, t_{\mathrm{ar}(\omega)})) < \infty$ for all $g \in \mathbf{dom}(\omega)$. Hence $\phi_M(t) < \infty$ for all $t \in \mathrm{Clone}^{(k)}(\bigcup_{\omega \in W} \mathbf{dom}(\omega))$. All other values of ϕ_M can be set to ∞, as this just reduces the number of inequalities in the system.

Set $C_k(W) = \mathrm{Clone}^{(k)}(\bigcup_{\omega \in W} \mathbf{dom}(\omega))$. Using translation of weighted operations (Definition 4.10), we can rewrite the inequalities (5) to obtain the following equivalent system: for all $\omega \in W$, and all $t_1, \ldots, t_{\mathrm{ar}(\omega)} \in C_k(W)$,

$$\sum_{h \in \mathbf{dom}(\omega[t_1, \ldots, t_{\mathrm{ar}(\omega)}])} \omega[t_1, \ldots, t_{\mathrm{ar}(\omega)}](h)\, \phi_M(h(x_1, \ldots, x_k)) \;\leq\; 0\,. \tag{7}$$

Now, by applying Lemma 5.1 to the resulting system of inequalities, we conclude that either a solution ϕ_M exists, in which case $\omega_0 \notin \mathrm{wPol}(\mathrm{Imp}(W))$, or else there exists a set of non-negative rational numbers

$$\{c_{\omega[t_1, \ldots, t_{\mathrm{ar}(\omega)}]} \mid \omega \in W, t_1, \ldots, t_{\mathrm{ar}(\omega)} \in C_k(W)\}$$

such that for every $f \in C_k(W)$,

$$\sum_{\omega \in W} \sum_{\substack{\langle t_1, \ldots, t_{\mathrm{ar}(\omega)} \rangle \\ t_i \in C_k(W)}} c_{\omega[t_1, \ldots, t_{\mathrm{ar}(\omega)}]} \omega[t_1, \ldots, t_{\mathrm{ar}(\omega)}](f) \geq \omega_0(f)\,. \tag{8}$$

By Definition 4.7, adding the left-hand side of these inequalities over all f gives 0, and so does adding the right hand sides, so each inequality must actually be an equality. In other words, ω_0 is equal to a positive weighted sum of translations of weighted operations in W.

Hence, by Lemma 5.3 and Remark 4.14, ω_0 is equal to a translation of some element $\omega'_0 \in \mathrm{wClone}(W)$, so $\omega_0 \in \mathrm{wClone}(W)$. □

6 Conclusions

We have presented an algebraic theory of valued constraint languages analogous to the theory of clones used to study the complexity of the classical constraint satisfaction problem. We showed that the complexity of any valued constraint language with rational costs is determined by certain algebraic properties of the cost functions allowed in the language: the weighted polymorphisms.

In previous work [35, 34] it has been shown that every tractable crisp constraint language can be characterised by an associated clone of operations. That work initiated the use of algebraic properties in the search for tractable constraint languages, an area that has seen considerable activity in recent years; see, for instance, [13, 11, 12, 9, 36, 26, 4, 1, 3, 5]. The results in this paper show that a similar result holds for the valued constraint satisfaction problem: every tractable valued constraint language is characterised by an associated weighted clone. We hope that our results here will provide a similar impetus for the investigation of tractable valued constraint satisfaction problems.

References

[1] L. Barto, M. Kozik, Constraint Satisfaction Problems of Bounded Width, in: Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS'09), 2009, pp. 461–471.

[2] L. Barto, M. Kozik, New Conditions for Taylor Varieties and CSP, in: Proceedings of the 25th IEEE Symposium on Logic in Computer Science (LICS'10), 2010, pp. 100-109.

[3] L. Barto, M. Kozik, M. Maróti, T. Niven, CSP dichotomy for special triads, Proceedings of the American Mathematical Society 137 (9) (2009) 2921–2934.

[4] L. Barto, M. Kozik, T. Niven, The CSP dichotomy holds for digraphs with no sources and no sinks (a positive answer to a conjecture of Bang-Jensen and Hell), SIAM Journal on Computing 38 (5) (2009) 1782–1802.

[5] J. Berman, P. Idziak, P. Marković, R. McKenzie, M. Valeriote, R. Willard, Varieties with few subalgebras of powers, Transactions of the American Mathematical Society 362 (3) (2010) 1445–1473.

[6] S. Bistarelli, H. Fargier, U. Montanari, F. Rossi, T. Schiex, G. Verfaillie, Semiring-based CSPs and Valued CSPs: Frameworks, Properties, and Comparison, Constraints 4 (3) (1999) 199–240.

[7] V. Bodnarčuk, L. Kalužnin, V. Kotov, B. Romov, Galois theory for Post algebras. I, Cybernetics and Systems Analysis 5 (3) (1969) 243–252.

[8] F. Börner, Basics of galois connections, in: Complexity of Constraints, vol. 5250 of Lecture Notes in Computer Science, Springer, 2008, pp. 38–67.

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

[10] A. Bulatov, A Graph of a Relational Structure and Constraint Satisfaction Problems, in: Proceedings of the 19th IEEE Symposium on Logic in Computer Science (LICS'04), 2004, pp. 448–457.

[11] A. A. Bulatov, Tractable Conservative Constraint Satisfaction Problems, in: Proceedings of the 18th IEEE Symposium on Logic in Computer Science (LICS'03), 2003, pp. 321–330.

[12] A. Bulatov, A. Krokhin, P. Jeavons, Classifying the Complexity of Constraints using Finite Algebras, SIAM Journal on Computing 34 (3) (2005) 720–742.

[13] A. Bulatov, A. Krokhin, P. Jeavons, The complexity of maximal constraint languages, in: Proceedings of the 33rd ACM Symposium on Theory of Computing (STOC'01), 2001, pp. 667–674.

[14] R. Burkard, B. Klinz, R. Rudolf, Perspectives of Monge Properties in Optimization, Discrete Applied Mathematics 70 (2) (1996) 95–161.

[15] D. Cohen, M. Cooper, P. Jeavons, A complete characterization of complexity for Boolean constraint optimization problems, in: Proceedings of the 10th International Conference on Constraint Programming (CP'04), vol. 3258 of Lecture Notes in Computer Science, Springer, 2004, pp. 212–226.

[16] D. A. Cohen, M. C. Cooper, P. G. Jeavons, An Algebraic Characterisation of Complexity for Valued Constraints, in: Proceedings of the 12th International Conference on Principles and Practice of Contraint Programming (CP'06), vol. 4204 of Lecture Notes in Computer Science, Springer, 2006, pp. 107–121.

[17] D. A. Cohen, M. C. Cooper, P. G. Jeavons, Generalising submodularity and Horn clauses: Tractable optimization problems defined by tournament pair multimorphisms, Theoretical Computer Science 401 (1-3) (2008) 36–51.

[18] D. Cohen, M. Cooper, P. Jeavons, A. Krokhin, A Maximal Tractable Class of Soft Constraints, Journal of Artificial Intelligence Research 22 (2004) 1–22.

[19] D. Cohen, M. Cooper, P. Jeavons, A. Krokhin, Soft Constraints: Complexity and Multimorphisms, in: Proceedings of the 9th International Conference on Constraint Programming (CP'03), vol. 2833 of Lecture Notes in Computer Science, Springer, 2003, pp. 244–258.

[20] D. Cohen, M. Cooper, P. Jeavons, A. Krokhin, Supermodular Functions and the Complexity of MAX-CSP, Discrete Applied Mathematics 149 (1-3) (2005) 53–72.

[21] D. A. Cohen, M. C. Cooper, P. G. Jeavons, A. A. Krokhin, The Complexity of Soft Constraint Satisfaction, Artificial Intelligence 170 (11) (2006) 983–1016.

[22] N. Creignou, A dichotomy theorem for maximum generalized satisfiability problems, Journal of Computer and System Sciences 51 (3) (1995) 511–522.

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

[24] W. Cunningham, Minimum cuts, modular functions,and matroid polyhedra, Networks 15 (2) (1985) 205–215.

[25] J.-C. A. d'Auriac, F. Igloi, M. Preismann, A. Sebö, Optimal cooperation and submodularity for computing Potts' partition functions with a large number of statistics, Journal of Physics A: Mathematical and General 35 (2002) 6973–6983.

[26] V. Deineko, P. Jonsson, M. Klasson, A. Krokhin, The approximability of Max CSP with fixed-value constraints, Journal of the ACM 55 (4).

[27] T. Feder, M. Vardi, The Computational Structure of Monotone Monadic SNP and Constraint Satisfaction: A Study through Datalog and Group Theory, SIAM Journal on Computing 28 (1) (1998) 57–104.

[28] S. Fujishige, Submodular Functions and Optimization, vol. 58 of Annals of Discrete Mathematics, 2nd ed., Elsevier, 2005.

[29] D. Geiger, Closed systems of functions and predicates, Pacific Journal of Mathematics 27 (1) (1968) 95–100.

[30] M. Gyssens, P. Jeavons, D. Cohen, Decomposing Constraint Satisfaction Problems Using Database Techniques, Artificial Intelligence 66 (1) (1994) 57–89.

[31] S. Iwata, Submodular Function Minimization, Mathematical Programming 112 (1) (2008) 45–64.

[32] S. Iwata, L. Fleischer, S. Fujishige, A combinatorial strongly polynomial algorithm for minimizing submodular functions, Journal of the ACM 48 (4) (2001) 761–777.

[33] S. Iwata, J. B. Orlin, A Simple Combinatorial Algorithm for Submodular Function Minimization, in: Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'09), 2009, pp. 1230–1237.

[34] P. Jeavons, On the Algebraic Structure of Combinatorial Problems, Theoretical Computer Science 200 (1-2) (1998) 185–204.

[35] P. Jeavons, D. Cohen, M. Gyssens, Closure Properties of Constraints, Journal of the ACM 44 (4) (1997) 527–548.

[36] P. Jonsson, M. Klasson, A. Krokhin, The Approximability of Three-valued MAX CSP, SIAM Journal on Computing 35 (6) (2006) 1329–1349.

[37] V. Kolmogorov, S. Živný, Generalising tractable VCSPs defined by symmetric tournament pair multimorphisms, Tech. rep., arXiv:1008.3104 (August 2010).

[38] V. Kolmogorov, S. Živný, The complexity of conservative finite-valued CSPs, Tech. rep., arXiv:1008.1555 (August 2010).

[39] H. Nagamochi, T. Ibaraki, Computing Edge-Connectivity in Multigraphs and Capacitated Graphs, SIAM Journal on Discrete Mathematics 5 (1) (1992) 54–66.

[40] H. Narayanan, A note on the minimization of symmetric and general submodular functions, Discrete Applied Mathematics 131 (2) (2003) 513–522.

[41] H. Narayanan, Submodular Functions and Electrical Networks, North-Holland, Amsterdam, 1997.

[42] G. Nemhauser, L. Wolsey, Integer and Combinatorial Optimization, John Wiley & Sons, 1988.

[43] J. B. Orlin, A faster strongly polynomial time algorithm for submodular function minimization., Mathematical Programming 118 (2) (2009) 237–251.

[44] M. Queyranne, Minimising symmetric submodular functions, Mathematical Programming 82 (1-2) (1998) 3–12.

[45] T. Schaefer, The Complexity of Satisfiability Problems, in: Proceedings of the 10th Annual ACM Symposium on Theory of Computing (STOC'78), 1978, pp. 216–226.

[46] A. Schrijver, A Combinatorial Algorithm Minimizing Submodular Functions in Strongly Polynomial Time, Journal of Combinatorial Theory, Series B 80 (2) (2000) 346–355.

[47] A. Schrijver, Theory of linear and integer programming, John Wiley & Sons, Inc., 1986.

[48] D. Topkis, Supermodularity and Complementarity, Princeton University Press, 1998.

[49] S. Živný, D. A. Cohen, P. G. Jeavons, The Expressive Power of Binary Submodular Functions, Discrete Applied Mathematics 157 (15) (2009) 3347–3358.

A Multimorphisms

In previous work [19, 15, 21, 17, 37] the algebraic notion of a *multimorphism* has been used to study the complexity of valued constraint languages. In this appendix, and the following one, we explain how multimorphisms are related to weighted polymorphisms, and show that a simple approach using multimorphisms does *not* establish a suitable Galois connection for valued constraints.

Definition A.1 ([19]). *A k-ary* **multi-function** *on D is a function $F : D^k \to D^k$. Each k-ary multi-function F corresponds to a tuple of functions $\langle f_1, \ldots, f_k \rangle$, where each $f_i : D^k \to D$.*

For any r-ary cost function ϕ, we say that a k-ary multi-function F is a **multimorphism** *of ϕ if, for all $x_1, \ldots, x_k \in D^r$,*

$$\sum_{i=1}^{k} \phi(x_i) \geq \sum_{i=1}^{k} \phi(f_i(x_1, \ldots, x_k)) .$$

For any set of cost functions Γ, we will say F is a multimorphism of Γ if F is a multimorphism of every $\phi \in \Gamma$. The set of all multimorphisms of Γ will be denoted $\mathrm{Mul}(\Gamma)$*.*

The submodular cost functions are characterised by having the binary multimorphism $\langle \min, \max \rangle$. In fact, every known tractable valued constraint language has been shown to be characterised by a single multimorphism [21], or two multimorphisms [37].

Definition A.2. *We call a multi-function F a* **multi-projection** *if there exists some permutation π on $\{1, 2, \ldots, k\}$, such that $F(x)[i] = x[\pi(i)]$ for each $i = 1, \ldots, k$.*

Note that every cost function has every multi-projection as a multimorphism.

Definition A.3. *Let $F = \langle f_1, \ldots, f_k \rangle$ and $G = \langle g_1, \ldots, g_k \rangle$ be a pair of k-ary multi-functions. We define the* **composition** *of F and G as follows:*

$$G \circ F = \langle f_1[g_1, \ldots, g_k], \ldots, f_k[g_1, \ldots, g_k] \rangle .$$

Definition A.4. *Let $F = \langle f_1, \ldots, f_k \rangle$ and $G = \langle g_1, \ldots, g_l \rangle$ be a pair of multi-functions. We define the* **addition** *of F and G as follows:*

$$F + G = \langle f_1(x_1, \ldots, x_k), \ldots, f_k(x_1, \ldots, x_k), g_1(x_{k+1}, \ldots, x_{k+l}), \ldots, g_l(x_{k+1}, \ldots, x_{k+l}) \rangle .$$

Definition A.5. *We call a set of multi-functions $\mathcal{F}$ a* **multi-clone** *if it contains all multi-projections and is closed under composition and addition.*

For any set of multi-functions $\mathcal{F}$, we define the multi-clone generated by $\mathcal{F}$, which we denote $\mathrm{MClone}(\mathcal{F})$*, to be the smallest multi-clone containing $\mathcal{F}$.*

It will be convenient to associate each multi-function with a weighted operation in the following way. Let $F = \langle f_1, \ldots, f_k \rangle$ be a k-ary multi-function, and define a corresponding weighted operation ω_F, with domain $\{f_1, f_2, \ldots, f_k\} \cup \mathbf{J}_D^{(k)}$, as follows:

$$\omega_F(f) \overset{\text{def}}{=} \begin{cases} \mathit{count}_F(f) - 1 & f = e_i^{(k)} \text{ for some } i = 1, \ldots, k \\ \mathit{count}_F(f) & f \in \{f_1, \ldots, f_k\} \end{cases}$$

where $count_F(f) = |\{i : f_i = f\}|$. It is straightforward to check that $F \in \mathrm{Mul}(\Gamma)$ if and only if $\omega_F \in \mathrm{wPol}(\Gamma)$.

By identifying F with ω_F, we can consider the multimorphisms of a valued constraint language Γ to be a (strict) subset of the weighted polymorphisms of Γ. This identification allows us to extend Definition 4.21 to sets of multi-functions, and hence define $\mathrm{Imp}(\mathcal{F})$ for any set of multi-functions $\mathcal{F}$.

A natural question to ask is whether $\mathrm{MClone}(\mathcal{F})$ is equal to $\mathrm{Mul}(\mathrm{Imp}(\mathcal{F}))$. If this were true, it might provide a simpler Galois connection than the one we have considered earlier in this paper.

It is straightforward to check that $\mathrm{MClone}(\mathcal{F}) \subseteq \mathrm{Mul}(\mathrm{Imp}(\mathcal{F}))$. However, in this appendix we will show that the reverse inclusion does not hold. More precisely, we will show that there exists a multi-function on a domain of size 2 which is a multimorphism of every submodular cost function, but which is not contained in $\mathrm{MClone}(\langle \min, \max \rangle)$. This result explains why it was necessary to introduce the more general notion of translation (Definition 4.10) in order to derive the Galois connection in Theorem 4.23.

To establish this result, we first derive certain structural properties of the elements of $\mathrm{MClone}(\langle \min, \max \rangle)$. In the following, we restrict our attention to the Boolean case and use the binary Boolean operations $x \wedge y$ and $x \vee y$ to represent $\min(x, y)$ and $\max(x, y)$, respectively. Since we are interested in multi-functions which are multimorphisms of all submodular functions, we can restrict our attention to elements of $\mathrm{Clone}(\wedge, \vee)$, i.e., the monotone Boolean functions. Any element $t \in \mathrm{Clone}^{(k)}(\wedge, \vee)$ can be represented as

$$t = \bigvee_{i \in I_t} \Big(\bigwedge_{j \in T_i} x_j \Big),$$

where I_t is a set of indices and $\{T_i : i \in I_t\}$ is a *Sperner family of subsets of* $\{1, \ldots, k\}$.[1]

Definition A.6. *The Boolean multi-function* SORT_i^j*, where* $j > i$*, is defined as follows:*

$$\mathrm{SORT}_i^j(x)[p] = \begin{cases} x[p] & \textit{if } p \notin \{i, j\} \\ x[i] \wedge x[j] & p = i \\ x[i] \vee x[j] & p = j \end{cases}$$

This operation sorts the values in positions i and j into ascending order.

The multi-function SORT_i^j can be seen as the multi-function obtained by adding the multi-function $\langle \min, \max \rangle$ and a $(k-2)$-ary multi-projection, and then composing with an appropriate multi-projection. Thus, we can view every element of $\mathrm{MClone}(\langle \min, \max \rangle)$ as a composition of some sequence of SORT_i^j functions and multi-projections. In fact, the next result shows that every element of $\mathrm{MClone}(\langle \min, \max \rangle)$ can be represented as a composition of SORT_i^j functions composed with a single multi-projection.

[1]A Sperner family of $\{1, \ldots, k\}$ is a set of subsets of $\{1, \ldots, k\}$ such that no set is contained in any other set.

Proposition A.7. *Every element of* $\mathrm{MClone}(\langle \min, \max \rangle)$ *can be represented as*

$$\mathrm{SORT}_{i_1}^{j_1} \circ \cdots \circ \mathrm{SORT}_{i_m}^{j_m} \circ \pi \,,$$

for some $i_1, j_1, \ldots, i_m, j_m$*, and some multi-projection* π*.*

Proof. By our discussion above, it is sufficient to show that any multi-function of the form $\pi \circ \mathrm{SORT}_i^j$ is equivalent to a multi-function of the form $\mathrm{SORT}_{i'}^{j'} \circ \pi'$. Hence in any composition sequence of SORT_i^j functions and multi-projections, the multi-projections can be moved to the end, and combined to form a single multi-projection.

Applying $\pi \circ \mathrm{SORT}_i^j$ to the tuple x yields the tuple

$$x_{\pi(1)}, \ldots, (x_{\pi(i)} \wedge x_{\pi(j)}), \ldots, (x_{\pi(i)} \vee x_{\pi(j)}), \ldots, x_{\pi(k)} \,.$$

There are two cases to consider. If $\pi(i) < \pi(j)$ then we can obtain the same sequence by applying $\mathrm{SORT}_{\pi(i)}^{\pi(j)} \circ \pi$.

On the other hand, if $\pi(i) > \pi(j)$ we must apply $\mathrm{SORT}_{\pi(j)}^{\pi(i)}$. We then need to apply the multi-projection obtained by composing π with the transposition $(i\ j)$ to get the same result as $\pi \circ \mathrm{SORT}_i^j$. □

Definition A.8. *We say* k*-ary multi-function* F *on* D *is* **conservative** *if, for all* $x \in D^k$ *the multi-sets of values* $\{x[i] : 1 \leq i \leq k\}$ *and* $\{F(x)[i] : 1 \leq i \leq k\}$ *are equal.*

Definition A.9. *We shall call a* k*-ary Boolean multi-function* F **lowering** *if, for each* $x \in \{0,1\}^k$ *we have that* $F(x)$ *is lexicographically less than or equal to* x*.*

It is straightforward to check that each function SORT_i^j is conservative and lowering, and the composition of any two conservative lowering functions is itself conservative and lowering.

Proposition A.10. *If a conservative multi-function* F *is lowering, and* π *is a permutation such that* $F \circ \pi \neq F$*, then* $F \circ \pi$ *is not lowering.*

Proof. Since π is not the identity, we know there must exist some value $i \in \{1, 2, \ldots, k\}$ such that $\pi(i) < i$. Let s be the smallest such value and let $x = 0^{s-1}1^{k-s+1}$. Since F is both conservative and lowering, we have $F(x) = x$. Applying π to x gives a tuple x' with $x'[\pi(s)] = 1$. Thus, since $\pi(s) < s$, we have $x' >_{lex} x$. □

We are now able to prove the main result of this appendix.

Theorem A.11. *There exists a multi-function which is a multimorphism of every submodular cost function, but which is not contained in* $\mathrm{MClone}(\langle \min, \max \rangle)$*.*

Proof. We prove this result by exhibiting a 4-ary multi-function F on a domain of size 2, which is a multimorphism of all submodular functions, but which is not equal to a composition of SORT functions.

We define $F = \langle f_1, f_2, f_3, f_4 \rangle$, where

$$\begin{aligned} f_1 &= x_1 \wedge x_3 \wedge x_4 \,, \\ f_2 &= (x_1 \wedge x_2 \wedge x_3) \vee (x_2 \wedge x_4) \,, \\ f_3 &= (x_1 \wedge x_2) \vee (x_1 \wedge x_4) \vee x_3 \,, \\ f_4 &= x_1 \vee x_2 \vee x_4 \,. \end{aligned}$$

Viewed as a table, we have

x_1	0000000011111111
x_2	0000111100001111
x_3	0011001100110011
x_4	0101010101010101
f_1	0000000000010001
f_2	0000010100000111
f_3	0011001101111111
f_4	0101111111111111

It is straightforward to check from this table that F is conservative and lowering. Hence by Proposition A.7 and Proposition A.10, F belongs to $\mathrm{MClone}(\langle \min, \max \rangle)$ if and only if it is equal to a composition of SORT functions.

Suppose $F = F' \circ \mathrm{SORT}_i^j$, where F' is obtained as the composition of SORT functions. Then, we must have $f_i < f_j$ (in the lattice of terms generated by $\wedge$ and $\vee$), and, moreover, there must be a pair $s, t \in \mathrm{Clone}^{(4)}(\vee, \wedge)$ such that $f_i = s \wedge t$ and $f_j = s \vee t$. The only pairs $\langle i, j \rangle$ that satisfy $f_i < f_j$ are $\langle 1, 3 \rangle$, $\langle 1, 4 \rangle$ and $\langle 2, 4 \rangle$.

Consider first the possibility that $i = 1$ and $j = 3$, and suppose $f_1 = s \wedge t$ and $f_3 = s \vee t$ for some $s, t \in \mathrm{Clone}^{(4)}(\vee, \wedge)$. We can write

$$\begin{aligned} s &= \bigvee_{i \in I_s} \left(\bigwedge_{j \in S_i} x_j \right) \\ t &= \bigvee_{i \in I_t} \left(\bigwedge_{j \in T_i} x_j \right) \,, \end{aligned}$$

where the sets $\{S_i : i \in I_s\}$ and $\{T_i : i \in I_t\}$ are both Sperner families of subsets of $\{1, \ldots, k\}$.

We must have each of $\{1, 2\}$, $\{1, 4\}$, $\{3\}$ equal to some S_i or T_i, since $f_3 = s \vee t$. Moreover, since $f_1 = x_1 \wedge x_3 \wedge x_4$, we can assume (w.l.o.g.) that $S_1 = \{1, 4\}$ and $T_1 = \{3\}$. Setting $S_2 = \{1, 2\}$ implies that

$$s \wedge t \geq (x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge x_3 \wedge x_4) > f_1 \,;$$

hence we cannot have $S_2 = \{1, 2\}$. On the other hand, setting $T_2 = \{1, 2\}$ gives

$$s \wedge t \geq (x_1 \wedge x_2 \wedge x_4) \vee (x_1 \wedge x_3 \wedge x_4) > f_1 \,.$$

Thus, we cannot have that $F = F' \circ \mathrm{SORT}^3_1$. A similar argument can be used to show that $F = F' \circ \mathrm{SORT}^4_1$ and $F = F' \circ \mathrm{SORT}^4_2$ are also both impossible.

Hence, we have shown that F cannot be generated as the composition of SORT functions, and hence cannot belong to $\mathrm{MClone}(\langle \min, \max \rangle)$.

Finally, we show that F is a multimorphism of all submodular functions. Let $W = \{\omega_{SM}\}$, where ω_{SM} is the weighted polymorphism associated with the multimorphism $\langle \min, \max \rangle$ that characterises the set of submodular cost functions (Example 4.19),

$$\omega_{SM}(f) = \begin{cases} -1 & \text{if } f \in \{e_1^{(2)}, e_2^{(2)}\} \\ +1 & \text{if } f \in \{x_1 \wedge x_2, x_1 \vee x_2\} \end{cases} .$$

Now consider the following 4 translations of ω_{SM}: $\omega_{SM}[x_i, y_i]$, where the y_i are given by

$$\begin{aligned} y_1 &= (x_1 \wedge x_3) \vee x_2 \vee x_4 \,, \\ y_2 &= (x_1 \wedge x_3) \vee x_4 \,, \\ y_3 &= (x_1 \wedge x_2) \vee (x_1 \wedge x_3) \vee (x_1 \wedge x_4) \,, \\ y_4 &= x_1 \wedge x_3 \,. \end{aligned}$$

Note that the following equalities hold:

$$\begin{aligned} x_1 \wedge y_1 &= y_3 \,, & x_1 \vee y_1 &= f_4 \,, \\ x_2 \wedge y_2 &= f_2 \,, & x_2 \vee y_2 &= y_1 \,, \\ x_3 \wedge y_3 &= y_4 \,, & x_3 \vee y_3 &= f_3 \,, \\ x_4 \wedge y_4 &= f_1 \,, & x_4 \vee y_4 &= y_2 \,. \end{aligned}$$

Thus, if we let $\omega' = \sum_{i=1}^4 \omega_{SM}[x_i, y_i]$ we have

$$\omega'(f) = \begin{cases} -1 & f \in \{x_i : 1 \leq i \leq 4\} \\ 0 & f \in \{y_i : 1 \leq i \leq 4\} \\ +1 & f \in \{f_i : 1 \leq i \leq 4\} \end{cases}$$

Since $\omega' \in \mathrm{wClone}(\omega_{SM})$, we know, by Theorem 4.23, that ω' is a weighted polymorphism of every cost function improved by ω_{SM}. Hence F satisfies the inequality in Defintion A.1 for all such cost functions, and so is a multimorphism of all submodular functions. $\square$

B Multimorphisms are not enough

As explained in Appendix A, the multimorphisms of a valued constraint language can be viewed as a subset of the set of weighted polymorphisms. This raises the natural question of whether the multimorphisms of a valued constraint language are sufficient in all cases to capture the expressive power of the language.

In this appendix we show that this is not true. We will call a weighted polymorphism ω *trivial* if $\omega(f) = 0$ for all $f \in \mathbf{dom}(\omega)$. We will construct a cost function, ϕ_D, with a non-trivial weighted polymorphism. However, we will also show that the only multimorphisms of ϕ_D are multi-projections. Hence, the expressive power of ϕ_D is not captured by considering its multimorphisms alone.

For any domain $D = \{1, 2, \ldots, d\}$ where $d = |D| > 2$ we define a binary cost function ϕ_D as follows:

$$\phi_D(x, y) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } x = 1 \text{ and } y = 2, \\ 1 & \text{if } x = 2 \text{ and } y = 1, \\ 2d^3 + x & \text{if } y = 2 \text{ and } x > 2, \\ 2d^3 + y & \text{if } x = 2 \text{ and } y > 2, \\ 2d^3 + 1/y & \text{if } x = 1 \text{ and } y > 2, \\ 2d^3 + 1/x & \text{if } y = 1 \text{ and } x > 2, \\ 2d^3 + d + 1 & \text{if } x = y, \\ 2d^3 & \text{otherwise.} \end{cases}$$

Theorem B.1. *The cost function ϕ_D has a non-trivial unary weighted polymorphism.*

Proof. For any $p, q \in D, p \neq q$, let $f_{p,q}$ be the unary function from D to D where

$$f_{p,q}(x) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } x = p > 2, \\ 2 & \text{if } x = q > 2, \\ x & \text{otherwise.} \end{cases}$$

Now consider the weighted operation $\omega \in \mathbf{W}_D^{(1)}$ defined as $\omega(e_1^{(1)}) = -1$ and $\omega(f_{p,q}) = 1/(d(d-1))$ for every $p \neq q$. We will prove that $\omega \in \mathrm{wPol}(\phi_D)$ by showing that ϕ_D and ω satisfy Inequality (3) from Definition 4.18, for every $\langle x, y\rangle \in D^2$.

If $\langle x, y\rangle \in \{1, 2\}^2$ then $f_{p,q}$ is the identity so

$$\frac{1}{d(d-1)} \sum_{p \neq q} \phi_D(f_{p,q}(x), f_{p,q}(y)) = \phi_D(x, y)\,.$$

For any $a \in D$, we have that $\phi_D(a, a) = 2d^3 + d + 1$. Hence, if $x = y$,

$$\frac{1}{d(d-1)} \sum_{p \neq q} \phi_D(f_{p,q}(x), f_{p,q}(y)) = \phi_D(x, y)\,.$$

Now consider $\langle x, y\rangle \notin \{1, 2\}^2$ satisfying $x \neq y$. In all such cases there is some $f_{p,q}$ for which $\phi_D(f_{p,q}(x), f_{p,q}(y)) \leq 1$; these are given in the table below.

$\langle x, y\rangle$	$\langle p, q\rangle$
$\langle 1, b\rangle$	$\langle 1, b\rangle$
$\langle 2, b\rangle$	$\langle b, 1\rangle$
$\langle a, 1\rangle$	$\langle 1, a\rangle$
$\langle a, 2\rangle$	$\langle a, 1\rangle$
$\langle a, b\rangle$	$\langle a, b\rangle$

Hence, for any $x, y \notin \{1,2\}^2$ satisfying $x \neq y$, the sum $\sum_{p \neq q} \phi_D(f_{p,q}(x), f_{p,q}(y))$ has at least one term with value ≤ 1. Then, since the maximum value of ϕ_D is $2d^3 + d + 1$, we have

$$\begin{aligned}
\frac{1}{d(d-1)} \sum_{p \neq q} \phi_D(f_{p,q}(x), f_{p,q}(y)) &\leq \left(1 - \frac{1}{d(d-1)}\right)(2d^3 + d + 1) + \frac{1}{d(d-1)} \\
&= 2d^3 - \frac{d^2+2}{d-1} \\
&\leq 2d^3 \\
&\leq \phi_D(x, y)\,.
\end{aligned}$$

The last inequality comes from the fact that $\phi_D(x, y) \geq 2d^3$ for all $x, y \in D^k$. $\square$

Next we show that every multimorphism of ϕ_D is a multi-projection. We first show that every multimorphism of ϕ_D must behave like a multi-projection on tuples from $\{1,2\}^k$.

Lemma B.2. *Any k-ary multimorphism of ϕ_D acts as a multi-projection when applied to tuples in $\{1,2\}^k$.*

Proof. Let F be a k-ary multimorphism ($k \geq 1$) of ϕ_D. We will first show that F is conservative on all tuples in $\{1,2\}^k$. Using this result, we are then able to show that F must behave like a multi-projection on $\{1,2\}^k$.

We start by observing that since $\langle 1, 2\rangle$ is the only assignment with cost 0, F must fix 1^k and 2^k. Next, suppose $x \in \{1,2\}^k$ and let $\langle x'_1, \ldots, x'_k\rangle = F(x)$. We will consider two tableau in our analysis of F: the $2 \times k$ tableau with i-th row equal to $\langle 1, x[i]\rangle$, and the $2 \times k$ tableau with i-th row $\langle x[i], 2\rangle$. By considering how F must behave on these two tableaux, we are able to show that F is conservative.

Suppose x has r 1s and $k - r$ 2s and x' has r' 1s, s' 2s, and t' values > 2. The total cost of applying ϕ_D to the rows in the first tableau is

$$\sum_{i=1}^{k} \phi_D(1, x[i]) = r(2d^3 + d + 1)\,, \tag{9}$$

and the corresponding cost for the tableau obtained by applying F to the first tableau is

$$\sum_{i=1}^{k} \phi_D(1, x'[i]) \geq r'(2d^3 + d + 1) + 2t'd^3\,. \tag{10}$$

Here we are using the fact that $\phi_D(a, b) \geq 2d^3$ when $a \neq b$ and $\langle a, b\rangle \notin \{1,2\}^2$. Since F is a multimorphism of ϕ_D, we must have (9) $\geq$ (10), by Definition A.1; this implies that

$$(r - r')(2d^3 + d + 1) \geq 2t'd^3\,. \tag{11}$$

Similarly, the total cost of applying ϕ_D to the rows in the second tableau is

$$\sum_{i=1}^{k} \phi_D(x[i],2) = (k-r)(2d^3+d+1)\,, \tag{12}$$

and the corresponding cost for the tableau obtained by applying F to the second tableau is

$$\sum_{i=1}^{k} \phi_D(x'[i],2) \geq s'(2d^3+d+1) + 2t'd^3\,. \tag{13}$$

Again, the fact that F is a multimorphism implies (12) $\geq$ (13); hence,

$$(k-r-s')(2d^3+d+1) \geq 2t'd^3\,. \tag{14}$$

Combining (11) and (14) gives

$$(k-r'-s')(2d^3+d+1) \geq 4t'd^3\,. \tag{15}$$

But $t' = k - r' - s'$, so (15) can only be true when $t' = 0$. Then (11) and (14) can only hold when $r = r'$. Hence, F is conservative on $\{1,2\}^k$.

Let v_i be the element of $\{1,2\}^k$ with exactly one 1 at position i. We define a permutation on $\{1,2,\ldots,k\}$ according to the action of F on the tuples v_i, by setting $\pi(i)$ to be the unique $j \in \{1,2,\ldots,k\}$ such that $F(v_i)[j] = 1$. We will now show that F behaves like the multi-projection defined by π, when applied to any tuple in $\{1,2\}^k$.

Since ϕ_D has maximum cost when its two arguments are equal it follows that if $x, y \in \{1,2\}^k$ have no 1s in common, i.e., there is no $i \in \{1,2,\ldots,k\}$ with $x[i] = y[i] = 1$, than $F(x)$ and $F(y)$ also have no 1s in common.

Now suppose $z \in \{1,2\}^k$ satisfies $z[i] = 2$ for some $i \in \{1,2,\ldots,k\}$. Then, since v_i and z have no 1s in common, it follows that $F(v_i)$ and $F(z)$ have no 1s in common, i.e., $F(z)[\pi(i)] = 2$. Since this holds for each i with $z[i] = 2$, we can conclude that F behaves like the permutation π when applied to any $z \in \{1,2\}^k$. $\square$

Theorem B.3. *The only multimorphisms of ϕ_D are multi-projections.*

Proof. Let F be a k-ary multimorphism of ϕ_D. By Lemma B.2, we know that there exists a permutation π on $\{1,2,\ldots,k\}$ such that F behaves like π when applied to tuples in $\{1,2\}^k$. Hence, if we compose F with the multi-projection corresponding to the inverse permutation, π^{-1}, we get a multimorphism F' of ϕ_D that behaves like the identity on $\{1,2\}^k$. We will now show that F' is, in fact, the identity. Then, since the inverse of any multi-projection is itself a multi-projection, we are able to conclude that F must be a multi-projection.

Let $x \in D^k$ and let $y = F'(x)$. We define a new tuple $z = z(x,y)$ according to the following table.

$\langle x[i], y[i] \rangle$	$z[i]$	$\phi_D(x[i], z[i])$	$\phi_D(y[i], z[i])$
$\langle 2, 1 \rangle$	1	1	$2d^3 + d + 1$
$\langle a, 1 \rangle$ where $a > 2$	1	$2d^3 + 1/a$	$2d^3 + d + 1$
$\langle 1, 2 \rangle$	2	0	$2d^3 + d + 1$
$\langle a, 2 \rangle$ where $a > 2$	2	$2d^3 + a$	$2d^3 + d + 1$
$\langle 1, b \rangle$ where $b > 2$	2	0	$2d^3 + b$
$\langle 2, b \rangle$ where $b > 2$	1	1	$2d^3 + 1/b$
$\langle a, b \rangle$ where $2 < a < b$	2	$2d^3 + a$	$2d^3 + b$
$\langle a, b \rangle$ where $a > b > 2$	1	$2d^3 + 1/a$	$2d^3 + 1/b$
$\langle a, a \rangle$	1	$\phi_D(a, 1)$	$\phi_D(a, 1)$

Consider the tableau whose columns are given by x and z. Since F' is a multimorphism of ϕ_D, we must have

$$\sum_{i=1}^{k} \phi_D(x[i], z[i]) \geq \sum_{i=1}^{k} \phi_D(y[i], z[i]) , \tag{16}$$

since $F'(z) = z$. As demonstrated in the above table, whenever $x[i] \neq y[i]$ we will have

$$\phi_D(x[i], z[i]) < \phi_D(y[i], z[i]) .$$

Thus, in order for (16) to hold, we must have $x[i] = y[i]$ for $i = 1, \ldots, k$. In other words, we must have $x = F'(x)$ for all $x \in D^k$. $\square$