The Expressive Power of Valued Constraints: Hierarchies and Collapses ⋆

David A. Cohen [a] Peter G. Jeavons [b] Stanislav Živný [b,*,1]

[a] *Dept. of Computer Science, Royal Holloway, University of London, UK*

[b] *Computing Laboratory, University of Oxford, UK*

Abstract

In this paper, we investigate the ways in which a fixed collection of valued constraints can be combined to express other valued constraints. We show that in some cases, a large class of valued constraints, of all possible arities, can be expressed by using valued constraints over the same domain of a fixed finite arity. We also show that some simple classes of valued constraints, including the set of all monotonic valued constraints with finite cost values, cannot be expressed by a subset of any fixed finite arity, and hence form an infinite hierarchy.

Key words: Valued constraint satisfaction, Expressibility, Max-closed cost functions, Polymorphisms, Feasibility polymorphisms, Fractional polymorphisms

1 Introduction

Building a computational model of a combinatorial problem means capturing the requirements and optimisation criteria of the problem, using the resources available in some given computational system. Modelling such problems using

⋆ A preliminary version of this paper appeared in *Proceedings of the 13th International Conference on Principles and Practice of Constraint Programming (CP)*, 2007, pp. 798-805.

* Corresponding author. Address: Computing Laboratory, University of Oxford, Wolfson Building, Parks Road, Oxford OX1 3QD, UK. Tel.: +44 (0)1865 273884. Fax: +44 (0)1865 273839.

Email addresses: dave@cs.rhul.ac.uk (David A. Cohen), peter.jeavons@comlab.ox.ac.uk (Peter G. Jeavons), stanislav.zivny@comlab.ox.ac.uk (Stanislav Živný).

[1] This work was supported by EPSRC grant EP/F01161X/1.

constraints means expressing the requirements and optimisation criteria, using some combination of basic constraints provided by the system. In this paper, we investigate what kinds of relations and functions can be expressed using a given set of allowed constraint types.

The classical constraint satisfaction problem (CSP) model considers only the feasibility of satisfying a collection of simultaneous requirements [26,9,30]. Various extensions have been proposed to this model, to allow it to deal with different kinds of optimisation criteria, or preferences between different feasible solutions. Two very general extended frameworks that have been proposed are the semi-ring CSP framework and the valued CSP (VCSP) framework [2].

The semi-ring framework is slightly more general [2], but the VCSP framework is simpler, and sufficiently powerful to describe many important classes of problems [30]. In particular, it generalises the classical CSP model, and includes many standard optimisation problems, such as MIN-CUT, MAX-SAT, MAX-ONES SAT and MAX-CSP [8].

In this paper, we work with the VCSP framework. In this framework every constraint has an associated cost function which assigns a cost to every tuple of values for the variables in the scope of the constraint. The set of cost functions used in the description of the problem is called the *valued constraint language*.

As with all computing paradigms, it is desirable for many purposes to have a small language which can be used to describe a large collection of problems. Determining which additional constraints can be *expressed* by a given valued constraint language is therefore a central issue in assessing the flexibility and usefulness of a constraint system, and it is this question that we investigate here.

The notion of *expressibility* has been a key component in the analysis of complexity for the classical CSP model [19,5]. It was also a major tool in the complexity analysis of a wide variety of Boolean constraint problems carried out by Creignou et al. [9], where it was referred to as *implementation*. Expressibility is a particular form of problem reduction: if a constraint can be expressed in a given constraint language, then it can be added to the language without changing the computational complexity of the associated class of problems. Hence determining what can be expressed in a given valued constraint language is a fundamental step in the complexity analysis of valued constraint problems.

[2] The main difference is that costs in VCSPs represent violation levels and have to be totally ordered, whereas costs in semi-ring CSPs represent preferences and might be ordered only partially.

In order to investigate the expressive power of valued constraint languages, we make use of a number of algebraic tools that have been developed for this question [22], and for the related question of determining the complexity of the associated constraint satisfaction problems [6,8]. By applying these tools to particular valued constraint languages, we show that some simple constraint classes provide infinite hierarchies of greater and greater expressive power, whereas other classes collapse to sets of cost functions of fixed arity which can express all the other cost functions in the class.

We remark on the relationship between our results and some previous work on the VCSP. Larrosa and Dechter showed [25] that both the so-called *dual* representation [11] and the *hidden variable* representation [10], which transform any CSP instance into a binary CSP instance, can be generalised to the VCSP framework. However, these representations involve an exponential blow-up (in the arity of the constraints) of the domain size (i.e., the set of possible values for each variable). The notion of expressibility that we are using in this paper always preserves the domain size. Our results clarify which cost functions can be expressed using a given valued constraint language over the same domain, by introducing additional (hidden) variables and constraints; the number of these that are required is fixed for any given cost function.

The paper is organised as follows. In Section 2, we define the standard valued constraint satisfaction problem and the notion of expressibility for valued constraints. In Section 3, we describe some algebraic techniques that have been developed for valued constraints in earlier papers and show how they can be used to investigate expressibility. In Section 4, we show that relations of a fixed arity can express any relation of any arbitrary arity. We show the same result for max-closed relations. In Section 5, we show that finite-valued cost functions of a fixed arity can express any finite-valued cost function of any arbitrary arity. By contrast, we show that the *finite-valued max-closed* cost functions form an infinite hierarchy. In other words, finite-valued max-closed cost functions of different arities have different expressive power. In Section 6, we show a collapse to finite arity for the set of all general cost functions taking both finite and infinite values. We show the same result for general max-closed cost functions. Finally in Section 7, we summarise our results and suggest some important open questions.

2 Valued Constraints and Expressibility

In this section, we define the valued constraint satisfaction problem, and discuss how the cost functions used to define valued constraints can be combined to express other valued constraints. A more detailed discussion of the valued constraint framework, and illustrative examples, can be found in [2,8].

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

- *V is a finite set of **variables**;*
- *D is a finite set of possible **values**;*
- *Ω is a valuation structure representing possible **costs**;*
- *$\mathcal{C}$ is a set of **valued constraints**. Each element of $\mathcal{C}$ is a pair $c = \langle \sigma, \phi \rangle$ where σ is a tuple of variables called the **scope** of c, and ϕ is a mapping from $D^{|\sigma|}$ to Ω, called the **cost function** of c.*

Definition 3 *For any VCSP instance $\mathcal{P} = \langle V, D, \mathcal{C}, \Omega \rangle$, 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,*

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

*A **solution** to $\mathcal{P}$ is an assignment with minimum cost.*

The complexity of finding an optimal solution to a valued constraint problem will obviously depend on the forms of valued constraints which are allowed in the problem [8]. In order to investigate different families of valued constraint problems, with different sets of allowed constraint types, we use the notion of a **valued constraint language**, which is simply a set of possible cost functions mapping D^k to Ω, for some fixed set D and some fixed valuation structure Ω. The class of all VCSP instances where the cost functions of the valued constraints are all contained in a valued constraint language Γ will be denoted VCSP(Γ).

In any VCSP instance, the variables listed in the scope of each valued constraint are *explicitly* constrained, in the sense that each possible combination of values for those variables is associated with a given cost. Moreover, if we choose *any* subset of the variables, then their values are constrained *implicitly* in the same way, due to the combined effect of the valued constraints. This motivates the concept of **expressibility** for cost functions, which is defined as follows:

Definition 4 *For any VCSP instance $\mathcal{P} = \langle V, D, \mathcal{C}, \Omega \rangle$, and any list $l = \langle v_1, \ldots, v_m \rangle$ of variables of $\mathcal{P}$, the **projection** of $\mathcal{P}$ onto l, denoted $\pi_l(\mathcal{P})$,*

is the m-ary cost function defined as follows:

$$\pi_l(\mathcal{P})(x_1, \ldots, x_m) \stackrel{\text{def}}{=} \min_{\{s:V\to D \mid \langle s(v_1),\ldots,s(v_m)\rangle = \langle x_1,\ldots,x_m\rangle\}} Cost_{\mathcal{P}}(s).$$

*We say that a cost function ϕ is **expressible** over a valued constraint language Γ if there exists an instance $\mathcal{P} \in \mathsf{VCSP}(\Gamma)$ and a list l of variables of $\mathcal{P}$ such that $\pi_l(\mathcal{P}) = \phi$. We call the pair $\langle \mathcal{P}, l\rangle$ a **gadget** for expressing ϕ over Γ.*

Showing that a cost function is expressible over a valued constraint language is a form of problem reduction: if ϕ is expressible over Γ, then there is a polynomial-time reduction from $\mathsf{VCSP}(\Gamma \cup \{\phi\})$ to $\mathsf{VCSP}(\Gamma)$, which is obtained by replacing each constraint involving ϕ with a suitable gadget.

In this paper we shall examine the expressibility of cost functions over three particular valuation structures which can be used to model a wide variety of problems [8]:

Definition 5 *Let Ω be a valuation structure and let $\phi : D^m \to \Omega$ be a cost function.*

- *If $\Omega = \{0, \infty\}$, then we call ϕ a **crisp** cost function.*
- *If $\Omega = \mathbb{Q}_+$, the set of non-negative rational numbers with the standard addition operation, $+$, then we call ϕ a **finite-valued** cost function.*
- *If $\Omega = \overline{\mathbb{Q}}_+$, the set of non-negative rational numbers together with infinity, with the standard addition operation (extended so that $a + \infty = \infty$, for every $a \in \overline{\mathbb{Q}}_+$), then we call ϕ a **general** cost function.*

Note that with any *relation* R over D we can associate a crisp cost function ϕ_R on D which maps tuples in R to 0 and tuples not in R to ∞. On the other hand, with any m-ary cost function ϕ we can associate a relation R_ϕ defined as $\langle x_1, \ldots, x_m\rangle \in R_\phi \Leftrightarrow \phi(x_1, \ldots, x_m) < \infty$, or equivalently an m-ary crisp cost function defined by:

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

In view of the close correspondence between crisp cost functions and relations we shall use these terms interchangeably in the rest of the paper.

For *crisp* cost functions (=relations) the notion of expressibility in Definition 4 corresponds precisely to the established notion of expressibility using conjunction and existential quantification (i.e., using *primitive positive formulas*) [5].

3 Expressive Power and Algebraic Properties

By Definition 1, adding a finite constant to any cost function leaves the ordering of the costs unchanged, and so has no effect on the set of solutions. Hence, for any valued constraint language Γ with costs in Ω, we define the **expressive power** of Γ, denoted $\langle\Gamma\rangle$, to be the set of all cost functions ϕ such that $\phi + \gamma$ is expressible over Γ for some constant $\gamma \in \Omega$ where $\gamma < \infty$.

A number of algebraic techniques to determine the expressive power of a given valued constraint language have been developed in earlier papers. To make use of these techniques, we first need to define some key terms.

The i-th component of a tuple t will be denoted by $t[i]$. Note that any operation on a set D can be extended to tuples over the set D in a standard way, as follows. For any function $f : D^k \to D$, and any collection of tuples $t_1, \ldots, t_k \in D^m$, define $f(t_1, \ldots, t_k) \in D^m$ to be the tuple $\langle f(t_1[1], \ldots, t_k[1]), \ldots, f(t_1[m], \ldots, t_k[m])\rangle$.

Definition 6 ([12]) *Let R be an m-ary relation over a finite set D and let f be a k-ary operation on D. Then f is a **polymorphism** of R if $f(t_1, \ldots, t_k) \in R$ for all choices of $t_1, \ldots, t_k \in R$.*

A valued constraint language, Γ, which contains only crisp cost functions (= relations) will be called a crisp constraint language. We will say that f is a polymorphism of a crisp constraint language Γ if f is a polymorphism of every relation in Γ. The set of all polymorphisms of Γ will be denoted $\mathsf{Pol}(\Gamma)$.

It follows from the results of [19] that the expressive power of a crisp constraint language is fully characterised by its polymorphisms:

Theorem 7 ([19]) *For any crisp constraint language Γ over a finite set*

$$R \in \langle\Gamma\rangle \;\Leftrightarrow\; \mathsf{Pol}(\Gamma) \subseteq \mathsf{Pol}(\{R\}).$$

Hence, a crisp cost function ϕ is expressible over a crisp constraint language Γ if and only if it has all the polymorphisms of Γ. See [21] for more on the connection between crisp constraint languages on the one hand, and clone theory and universal algebra on the other hand.

We can extend the idea of polymorphisms to arbitrary valued constraint languages by considering the corresponding feasibility relations:

Definition 8 ([6]) *The **feasibility polymorphisms** of a valued constraint language Γ are the polymorphisms of the corresponding crisp feasibility cost functions, that is,* $\mathsf{FPol}(\Gamma) \stackrel{\text{def}}{=} \mathsf{Pol}(\{\mathsf{Feas}(\phi) \mid \phi \in \Gamma\})$.

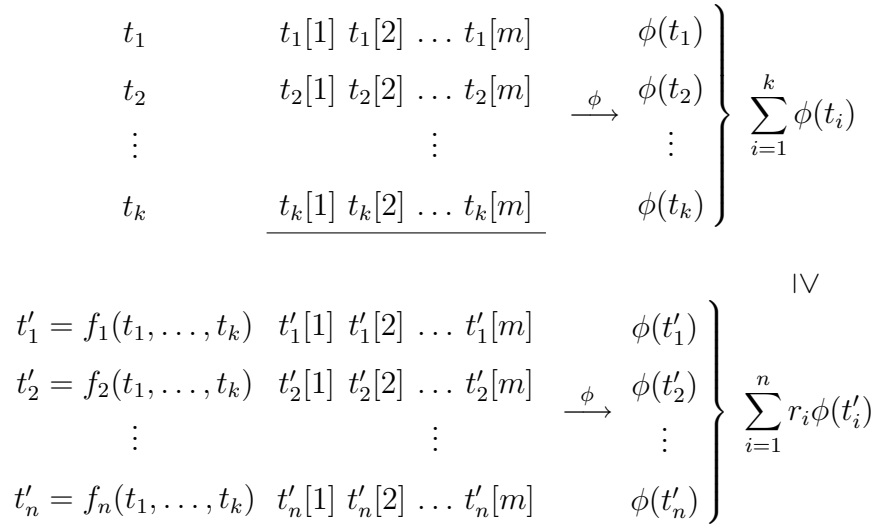

Figure 1. Definition of a fractional polymorphism $\mathcal{F} = \{\langle r_1, f_1\rangle, \ldots, \langle r_n, f_n\rangle\}$.

However, to fully capture the expressive power of valued constraint languages it is necessary to consider more general algebraic properties, such as the following (see Figure 1 for an illustration of Definition 9).

Definition 9 ([6]) *A k-ary **weighted function** $\mathcal{F}$ on a set D is a set of the form $\{\langle r_1, f_1\rangle, \ldots, \langle r_n, f_n\rangle\}$ where each r_i is a non-negative rational number such that $\sum_{i=1}^{n} r_i = k$ and each f_i is a distinct function from D^k to D.*

*For any m-ary cost function ϕ, we say that a k-ary weighted function $\mathcal{F}$ is a k-ary **fractional polymorphism** of ϕ if, for all $t_1, \ldots, t_k \in D^m$,*

$$\sum_{i=1}^{k} \phi(t_i) \;\geq\; \sum_{i=1}^{n} r_i \phi(f_i(t_1, \ldots, t_k)).$$

For any valued constraint language Γ, we will say that $\mathcal{F}$ is a fractional polymorphism of Γ if $\mathcal{F}$ is a fractional polymorphism of every cost function in Γ. The set of all fractional polymorphisms of Γ will be denoted $\mathsf{fPol}(\Gamma)$.

It is a simple consequence of the definitions that if $\{f_i\}_{1\leq i\leq n}$ are polymorphisms of a relation R, then any weighted function $\{\langle r_1, f_1\rangle, \ldots, \langle r_n, f_n\rangle\}$ is a fractional polymorphism of the corresponding crisp cost function ϕ_R. Conversely, if $\{\langle r_1, f_1\rangle, \ldots, \langle r_n, f_n\rangle\}$ is a fractional polymorphism of ϕ, then $\{f_i\}_{1\leq i\leq n}$ are polymorphisms of the corresponding relation R_ϕ.

It was shown in [6] that the feasibility polymorphisms and fractional polymorphisms of a valued constraint language effectively determine its expressive power. One consequence of this result is the following theorem:

Theorem 10 ([6]) *If Γ is a valued constraint language with costs in $\overline{\mathbb{Q}}_+$ such*

that, for all $\phi \in \Gamma$*, and all* $c \in \mathbb{Q}_+$*,* $c\phi \in \Gamma$ *and* $\mathsf{Feas}(\phi) \in \Gamma$*, then*

$$\phi \in \langle\Gamma\rangle \;\Leftrightarrow\; \mathsf{FPol}(\Gamma) \subseteq \mathsf{FPol}(\{\phi\}) \wedge \mathsf{fPol}(\Gamma) \subseteq \mathsf{fPol}(\{\phi\}).$$

Hence, for all valued constraint languages Γ satisfying the conditions of Theorem 10, a cost function ϕ is expressible over Γ if and only if it has all the feasibility polymorphisms of Γ and all the fractional polymorphisms of Γ.

4 The Expressive Power of Arbitrary Relations and Max-Closed Relations

In this section, we consider the expressive power of valued constraint languages containing only *crisp* cost functions, that is, *relations*.

We consider the languages containing all relations up to some fixed arity over some fixed domain, and we also consider an important subset of these relations defined for totally ordered domains, the so-called *max-closed* relations, which are defined below. In both cases, we show that the relations of a fixed arity can express all relations of arbitrary arities.

Definition 11 *Let* D *be a fixed totally ordered set.*

- *The* k*-ary function on* D *which returns the largest of its* k *arguments in the given ordering of* D *is denoted* $\textsc{Max}_k$*.*
- *The* k*-ary function on* D *which returns the smallest of its* k *arguments in the given ordering of* D *is denoted* $\textsc{Min}_k$*.*
- *The* k*-ary function on* D *which returns the second largest of its* $k \geq 2$ *arguments in the given ordering of* D *is denoted* $\textsc{Second}_k$*.*

The function $\textsc{Max}_2$ will be denoted Max and the function $\textsc{Min}_2$ will be denoted Min.

Definition 12 *A cost function* ϕ *is called* ***max-closed*** *if* $\{\langle 2, \textsc{Max}\rangle\} \in \mathsf{fPol}(\{\phi\})$.

In this section, we focus on *crisp* max-closed cost functions. This class of cost functions was first introduced (as a class of relations) in [23] and shown to be tractable. In other words, $\mathsf{VCSP}(\Gamma)$ is known to be polynomial-time solvable for any set Γ consisting of max-closed relations over any finite set D. A number of examples of max-closed relations are given in [23].

Definition 13 *For every* $d \geq 2$ *we define the following:*

- $\mathbf{R}_{d,m}$ *denotes the set of all relations of arity at most* m *over a domain of size* d*, and* $\mathbf{R}_d \stackrel{\text{def}}{=} \cup_{m\geq 0}\mathbf{R}_{d,m}$*;*
- $\mathbf{R}^{\max}_{d,m}$ *denotes the set of all max-closed relations of arity at most* m *over an ordered domain of size* d*, and* $\mathbf{R}^{\max}_d \stackrel{\text{def}}{=} \cup_{m\geq 0}\mathbf{R}^{\max}_{d,m}$.

It is well-known that any relation can be expressed as a propositional formula in conjunctive normal form (CNF), hence we have the following characterisation of $\mathbf{R}_{d,m}$.

Proposition 14 *A relation* $R \in \mathbf{R}_{d,m}$ *if and only if there is some formula* ψ *such that* $\langle v_1, \ldots, v_m\rangle \in R \Leftrightarrow \psi(v_1, \ldots, v_m)$ *and* ψ *is a conjunction of clauses of the form* $(v_1 \neq a_1) \vee \ldots \vee (v_m \neq a_m)$ *for some constants* $a_1, \ldots, a_m$.

We also have a similar characterisation for $\mathbf{R}^{\max}_{d,m}$, adapted from Theorem 5.2 of [23].

Theorem 15 ([23]) *A relation* $R \in \mathbf{R}^{\max}_{d,m}$ *if and only if there is some formula* ψ *such that* $\langle v_1, \ldots, v_m\rangle \in R \Leftrightarrow \psi(v_1, \ldots, v_m)$ *and* ψ *is a conjunction of clauses of the form* $(v_1 > a_1) \vee \ldots \vee (v_m > a_m) \vee (v_i < b_i)$ *for some constants* $a_1, \ldots, a_m, b_i$.

Note that in the special case of a Boolean domain (that is, when $d = 2$) this restricted form of clauses is equivalent to a disjunction of literals with at most one negated literal; clauses of this form are sometimes called **anti-Horn** clauses.

It is well-known that for every $d \geq 2$, $\mathsf{Pol}(\mathbf{R}_d)$ is equal to the set of all possible projection operations [12]. We now characterise the polymorphisms of $\mathbf{R}^{\max}_d$.

Definition 16 *Let* $I = \{i_1, \ldots, i_n\} \subseteq \{1, \ldots, k\}$ *be a set of indices. Define the* k*-ary function*

$$\textsc{Max}_I(x_1, \ldots, x_k) \stackrel{\text{def}}{=} \textsc{Max}_n(x_{i_1}, \ldots, x_{i_n}).$$

For every k, there are exactly $2^k - 1$ functions of the form $\textsc{Max}_I$ for $\emptyset \neq I \subseteq \{1, \ldots, k\}$.

Proposition 17 *For all* $d \geq 2$*,*

$$\mathsf{Pol}(\mathbf{R}^{\max}_d) = \{\textsc{Max}_I \mid \emptyset \neq I \subseteq \{1, \ldots, k\}, k = 1, 2, \ldots\}.$$

PROOF. When $|I| = 1$, the corresponding function $\textsc{Max}_I$ is just a projection operation, and every projection is a polymorphism of every relation [12].

If $\textsc{Max} \in \mathsf{Pol}(\{R\})$, then $\textsc{Max}_I \in \mathsf{Pol}(\{R\})$ for every $\emptyset \neq I \subseteq \{1, \ldots, k\}$. This is because $\mathsf{Pol}(\{R\})$ is closed under function composition and contains all projection operations, and every $\textsc{Max}_I$ can be obtained by function composition from the function Max and the projection operations.

We now prove that the operations of the form $\textsc{Max}_I$ are the *only* polymorphisms of $\mathbf{R}_d^{\max}$. Suppose, for contradiction, that f is a k-ary polymorphism of $\mathbf{R}_d^{\max}$ which is different from $\textsc{Max}_I$ for every $\emptyset \neq I \subseteq \{1, \ldots, k\}$. It follows that, for each I such that $\emptyset \neq I \subseteq \{1, \ldots, k\}$, there is a k-tuple t_I, such that $f(t_I) \neq \textsc{Max}_I(t_I)$. Let n be the total number of different tuples t_I, that is, $n = |\{t_I \mid \emptyset \neq I \subseteq \{1, \ldots, k\}\}| \leq 2^k - 1$ and denote these tuples by $t_1, \ldots, t_n$. Now consider the n-ary relation $R = \{\langle t_1[j], \ldots, t_n[j]\rangle\}_{1 \leq j \leq k}$. Define $R_0 = R$ and $R_{i+1} = R_i \cup \{\textsc{Max}(u, v) \mid u, v \in R_i\}$ for every $i \geq 0$. Clearly, $R_i \subseteq R_{i+1}$ and since there is only a finite number of different k-tuples, there is an l such that $R_l = R_{l+i}$ for every $i \geq 0$. Define R' to be the closure of R under Max, that is, $R' = R_l$. Clearly, R' is max-closed and every tuple t of R' is of the form $t = \textsc{Max}_j(u_{i_1}, \ldots, u_{i_j})$ for some $j \geq 1$ and $u_{i_1}, \ldots, u_{i_j} \in R$. We have constructed R so that the application of f to the tuples of R results in a tuple t which is different from every tuple of this form, and hence $t \notin R'$. Therefore, $f \notin \mathsf{Pol}(R')$, which means that $f \notin \mathsf{Pol}(\mathbf{R}_d^{\max})$.

We now consider the expressive power of $\mathbf{R}_{d,m}$ and $\mathbf{R}_{d,m}^{\max}$.

It is clear that binary relations have greater expressive power than unary relations, so our first result is not unexpected, but it provides a simple illustration of the use of the algebraic approach.

Proposition 18 *For all $d \geq 2$, $\langle \mathbf{R}_{d,1} \rangle \subsetneq \langle \mathbf{R}_{d,2} \rangle$ and $\langle \mathbf{R}_{d,1}^{\max} \rangle \subsetneq \langle \mathbf{R}_{d,2}^{\max} \rangle$.*

PROOF. Notice for example that $\textsc{Min} \in \mathsf{Pol}(\mathbf{R}_{d,1})$ and consequently $\textsc{Min} \in \mathsf{Pol}(\mathbf{R}_{d,1}^{\max})$ but $\textsc{Min} \notin \mathsf{Pol}(\mathbf{R}_{d,2})$ and $\textsc{Min} \notin \mathsf{Pol}(\mathbf{R}_{d,2}^{\max})$. The result then follows from Theorem 7.

4.1 Relations over a Boolean domain

As a first step, we now focus on the special case of relations over a Boolean domain, that is, the case when $d = 2$. This special case has been studied in detail in [3]. Here, we give a brief independent derivation of the relevant results using the techniques introduced above. We first show that the set of all ternary relations over a Boolean domain has fewer polymorphisms than the set of all binary relations, and hence has a greater expressive power. We also establish similar results for max-closed relations over a Boolean domain.

Proposition 19 MAJORITY $\in \mathsf{Pol}(\mathbf{R}_{2,2})$ *and* MAJORITY $\in \mathsf{Pol}(\mathbf{R}^{\max}_{2,2})$*, where* MAJORITY *is the unique ternary function on a 2-element set which returns the argument value that occurs most often.*

PROOF. Let R be an arbitrary binary Boolean relation. Let $a = \langle a_1, a_2\rangle$, $b = \langle b_1, b_2\rangle$ and $c = \langle c_1, c_2\rangle$ be three pairs belonging to R. Note that since the domain size is 2, the pair $\langle \textsc{Majority}(a_1, b_1, c_1), \textsc{Majority}(a_2, b_2, c_2)\rangle$ is equal to at least one of a, b, c, and hence belongs to R.

Proposition 20 MAJORITY $\notin \mathsf{Pol}(\mathbf{R}_{2,3})$ *and* MAJORITY $\notin \mathsf{Pol}(\mathbf{R}^{\max}_{2,3})$.

PROOF. Consider the ternary Boolean max-closed relation R consisting of all triples except $\langle 0, 0, 0\rangle$. To see that MAJORITY is not a polymorphism of R, consider the triples $\langle 0, 0, 1\rangle$, $\langle 0, 1, 0\rangle$ and $\langle 1, 0, 0\rangle$. The application of MAJORITY to these tuples results in the triple $\langle 0, 0, 0\rangle$ which is not in R.

However, we now show that *ternary* Boolean relations have the same expressive power as *all* Boolean relations. In other words, any Boolean relation of arbitrary arity is expressible by relations of arity at most three. The same result also holds for max-closed Boolean relations.

Proposition 21 $\mathbf{R}_2 \subseteq \langle \mathbf{R}_{2,3}\rangle$ *and* $\mathbf{R}^{\max}_2 \subseteq \langle \mathbf{R}^{\max}_{2,3}\rangle$.

PROOF. By Proposition 14, any Boolean relation $R \in \mathbf{R}_2$ can be expressed as a CNF formula ψ. By the standard SATISFIABILITY to 3-SATISFIABILITY reduction [15], there is a 3-CNF formula ψ' expressing R such that ψ is satisfiable if and only if ψ' is satisfiable.

Since the standard SATISFIABILITY to 3-SATISFIABILITY reduction preserves the anti-Horn form of clauses, the same result holds for max-closed Boolean relations.

Combining these results with Theorem 7, we obtain the following result.

Theorem 22

(1) $\langle \mathbf{R}_{2,1}\rangle \subsetneq \langle \mathbf{R}_{2,2}\rangle \subsetneq \langle \mathbf{R}_{2,3}\rangle = \mathbf{R}_2$*;*
(2) $\langle \mathbf{R}^{\max}_{2,1}\rangle \subsetneq \langle \mathbf{R}^{\max}_{2,2}\rangle \subsetneq \langle \mathbf{R}^{\max}_{2,3}\rangle = \mathbf{R}^{\max}_2$.

4.2 Relations over larger domains

For relations over a domain with 3 or more elements, similar results can be obtained. In fact, in this case we show that *any* relation can be expressed using *binary* relations.

Proposition 23 *For all $d \geq 3$, $\mathbf{R}_d \subseteq \langle \mathbf{R}_{d,2} \rangle$.*

PROOF. Without loss of generality, assume that $D = \{0, \ldots, M\}$, where $M = d - 1$. Define the binary relation R_d by

$$R_d = \{\langle 0, i \rangle, \langle i, 0 \rangle \mid 0 \leq i \leq M\} \cup \{\langle i, i+1 \rangle \mid 1 \leq i < M\}.$$

It is known that the only polymorphisms of the relation R_d are projection operations [13]. Hence, by Theorem 7, $\langle \{R_d\} \rangle = \mathbf{R}_d$.

A constructive proof of Proposition 23 can be found in [33].

By investigating the polymorphisms of binary max-closed relations, we now show that max-closed relations over non-Boolean domains can also be expressed using binary relations.

Theorem 24 *For all $d \geq 3$, $\mathbf{R}_d^{\max} \subseteq \langle \mathbf{R}_{d,2}^{\max} \rangle$.*

PROOF. We will show that $\mathsf{Pol}(\mathbf{R}_{d,2}^{\max}) \subseteq \mathsf{Pol}(\mathbf{R}_d^{\max})$. The result then follows from Theorem 7.

Without loss of generality, assume that $D = \{0, \ldots, M\}$ where $M = d - 1$. Let $f \in \mathsf{Pol}(\mathbf{R}_{d,2}^{\max})$ be an arbitrary k-ary polymorphism. By Proposition 17, it is enough to show that $f = \textsc{Max}_I$ for some $\emptyset \neq I \subseteq \{1, \ldots, k\}$.

First note that for any subset $S \subseteq D$, the binary relation $R = \{\langle a, a \rangle \mid a \in S\}$ is max-closed, so $f(x_1, \ldots, x_k) \in \{x_1, \ldots, x_k\}$. In other words, f is conservative.

If $f = \textsc{Max}_{\{1,\ldots,k\}}$ we are done. Otherwise, there exist $a_1, \ldots, a_k \in D$ such that $a_i = \textsc{Max}_k(a_1, \ldots, a_k)$ and $a_i > f(a_1, a_2, \ldots, a_k) = a_j$. Without loss of generality, in order to simplify our notation, assume that $i = 1$ and $j = 2$, that is, $a_1 = \textsc{Max}_k(a_1, \ldots, a_k)$ and $a_1 > f(a_1, a_2, \ldots, a_k) = a_2$. We will show that f does not depend on its first parameter.

For any fixed $x_2, \ldots, x_k \in D$, we denote the tuple $\langle x_2, \ldots, x_k \rangle$ by $\bar{x}$, and we define the binary max-closed relation

$$R_{\bar{x}} = (\{a_2, \ldots, a_k\} \times \{x_2, \ldots, x_k\}) \cup (\{a_1\} \times D).$$

Now consider the function $g_{\bar{x}}(r) = f(r, x_2, \ldots, x_k)$. Note that $g_{\bar{x}}(r)$ is a restriction of f with all arguments except the first one fixed.

Claim 1 $\forall r \in D,\ g_{\bar{x}}(r) \in \{x_2, \ldots, x_k\}$.

To establish this claim, note that for all $r \in D$ we have $\langle a_1, r \rangle \in R_{\bar{x}}$, and $\{\langle a_j, x_j \rangle \mid j = 2, \ldots, k\} \subseteq R_{\bar{x}}$. Since f is a polymorphism of $R_{\bar{x}}$ and $f(a_1, a_2, \ldots, a_k) = a_2$, it follows from the definition of $R_{\bar{x}}$ that $g_{\bar{x}}(r) \in \{x_2, \ldots, x_k\}$.

Now we show that if the largest element of the domain, M, is not among $x_2, \ldots, x_k$, then $g_{\bar{x}}(r)$ is constant.

Claim 2 $M \notin \{x_2, \ldots, x_k\} \Rightarrow \forall r \in D,\ g_{\bar{x}}(r) = g_{\bar{x}}(M)$.

To establish this claim, define the binary max-closed relation

$$R'_{\bar{x}} = (\{M\} \times D) \cup \{\langle x_j, x_j \rangle \mid j = 2, \ldots, k\}.$$

For all $r \in D$ we have $\langle M, r \rangle \in R'_{\bar{x}}$ and $\{\langle x_j, x_j \rangle \mid j = 2, \ldots, k\} \subseteq R'_{\bar{x}}$. By Claim 1, $g_{\bar{x}}(M) = x_i$ for some $2 \leq i \leq k$. Since f is a polymorphism of $R'_{\bar{x}}$, it follows from the definition of $R_{\bar{x}}$ that $g_{\bar{x}}(r) = x_i = g_{\bar{x}}(M)$ for every $r \in D$.

Next we generalise Claim 2 to show that $g_{\bar{x}}(r)$ is constant whenever $x_2, \ldots, x_k$ does not contain all elements of the domain D.

Claim 3 $\{x_2, \ldots, x_k\} \neq D \Rightarrow \forall r \in D,\ g_{\bar{x}}(r) = g_{\bar{x}}(M)$.

To establish this claim, we will show that for every $p \in D$, if $p \notin \{x_2, \ldots, x_k\}$, then $g_{\bar{x}}(r) = g_{\bar{x}}(M)$ for every $r \in D$. Note that the case $p = M$ is already proved by Claim 2. For any $p \in D \setminus \{M\}$, define the binary max-closed relation $R_p = \{\langle d, \Delta_p(d) \rangle \mid d \in D\}$, where

$$\Delta_p(x) = \begin{cases} x & \text{if } x \leq p, \\ x - 1 & \text{if } x > p. \end{cases}$$

For all $r \in D$ we have $\langle r, \Delta_p(r) \rangle \in R_p$ and $\{\langle x_j, \Delta_p(x_j) \rangle \mid j = 2, \ldots, k\} \subseteq R_p$. Since f is a polymorphism of R_p, it follows from the definition of R_p that for every $r \in D$, $g_{\bar{x}}(r) \in \Delta_p^{-1}(g_{\Delta_p(\bar{x})}(\Delta_p(r)))$.

Since $M \notin \{\Delta_p(d) \mid d \in D\}$, we know, by Claim 2, that $g_{\Delta_p(\bar{x})}(\Delta_p(r))$ is constant. Say $g_{\Delta_p(\bar{x})}(\Delta_p(r)) = k_p$. If $k_p \neq p$, then $|\Delta_p^{-1}(k_p)| = 1$ and so $g_{\bar{x}}$ is

constant. Alternatively, if $k_p = p$, then $\Delta_p^{-1}(k_p) = \{p, p+1\}$. In this case if $p \notin \{x_2, \ldots, x_k\}$, then we know, by Claim 1, that $g_{\bar{x}}(r) \neq p$, so $g_{\bar{x}}$ is again constant. This completes the proof of Claim 3.

Claim 4 *$g_{\bar{x}}(r)$ is constant.*

To establish this claim, define the binary max-closed relations $R_+ = \{\langle d, \Delta_+(d)\rangle \mid d \in D\}$ and $R_- = \{\langle d, \Delta_-(d)\rangle \mid d \in D\}$, where

$$\Delta_+(x) = \begin{cases} x & \text{if } x \neq M, \\ x-1 & \text{if } x = M \end{cases}$$

and

$$\Delta_-(x) = \begin{cases} x & \text{if } x \neq 0, \\ x+1 & \text{if } x = 0. \end{cases}$$

Define $\bar{y} = \langle \Delta_+(x_2), \ldots, \Delta_+(x_k)\rangle$ and $\bar{z} = \langle \Delta_-(x_2), \ldots, \Delta_-(x_k)\rangle$. Since $M \notin \{\Delta_+(d) \mid d \in D\}$ and $0 \notin \{\Delta_-(d) \mid d \in D\}$, we know, by Claim 3, that $g_{\bar{y}}$ and $g_{\bar{z}}$ are both constant.

For every $r \in D$, $\langle r, \Delta_+(r)\rangle \in R_+$ and for every $i = 2, \ldots, k$, $\langle x_i, \Delta_+(x_i)\rangle \in R_+$. Since f is a polymorphism of R_+, and $g_{\bar{y}}$ is constant, $g_{\bar{x}}$ is either constant or for every $r \in D$, $g_{\bar{x}}(r) \in \{M, M-1\}$. Similarly, for every $r \in D$, $\langle r, \Delta_-(r)\rangle \in R_-$ and for every $i = 2, \ldots, k$, $\langle x_i, \Delta_-(x_i)\rangle \in R_-$. Since f is a polymorphism of R_-, and $g_{\bar{z}}$ is constant, $g_{\bar{x}}$ is either constant or for every $r \in D$, $g_{\bar{x}}(r) \in \{0, 1\}$. Since $|D| > 2$ we know [3] that $|\{M, M-1\} \cap \{0, 1\}| \leq 1$. Hence, in all cases $g_{\bar{x}}$ is constant.

We have shown that if $a_1 = \max(a_1, \ldots, a_k)$ and $f(a_1, \ldots, a_k) < a_1$, then f does not depend on its first parameter. Similarly, by repeating the same argument, we can show that if $f \neq \textsc{Max}_{\{2,\ldots,k\}}$, then f does not depend on its i-th parameter for some i such that $2 \leq i \leq k$. Moreover, further repeating the same argument shows that if f does not depend on any parameter outside of $I \subseteq \{1, \ldots, k\}$ and $f \neq \textsc{Max}_I$, then f does not depend on any of the parameters whose index is in I.

Therefore, either there is some set $I \subseteq \{1, \ldots, k\}$ for which $f = \textsc{Max}_I$ or else f is constant. However, since f is conservative, it cannot be constant.

Combining these results we obtain the following result:

Theorem 25 *For all $d \geq 3$,*

[3] This is the only place where we use the condition that $|D| \geq 3$.

(1) $\langle \mathbf{R}_{d,1} \rangle \subsetneq \langle \mathbf{R}_{d,2} \rangle = \mathbf{R}_d$*;*
(2) $\langle \mathbf{R}^{\max}_{d,1} \rangle \subsetneq \langle \mathbf{R}^{\max}_{d,2} \rangle = \mathbf{R}^{\max}_d$.

Figure 2 summarises the results from this section.

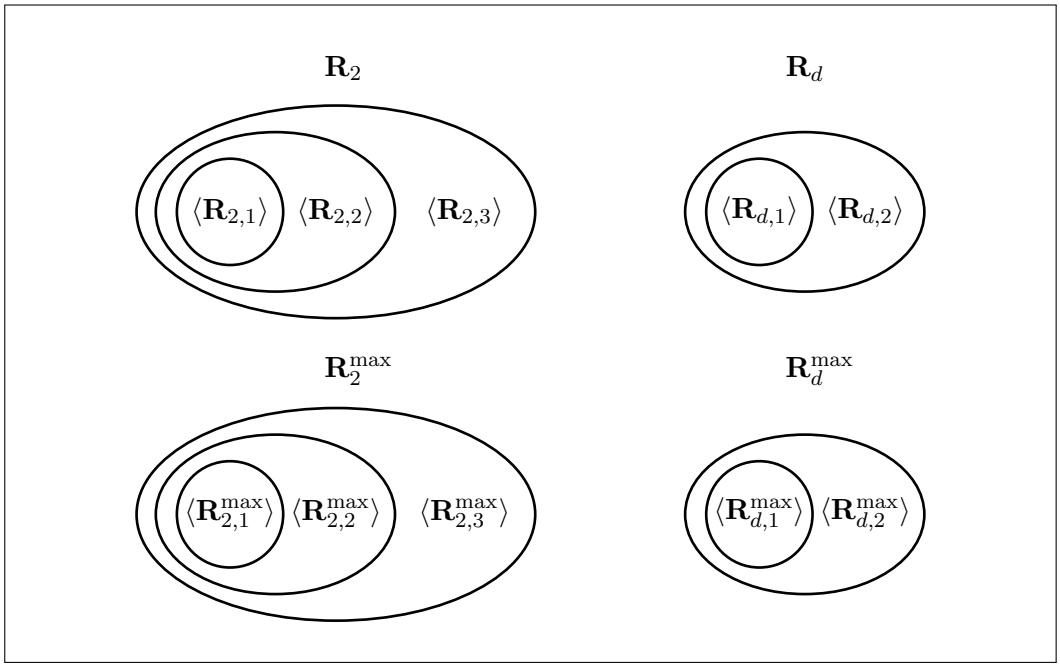

Figure 2. Summary of results from Section 4, for all $d \geq 3$.

5 Finite-valued Cost Functions

In this section, we consider the expressive power of valued constraint languages containing only *finite-valued* cost functions. First we show that the set of all finite-valued cost functions of a certain fixed arity can express all finite-valued cost functions of arbitrary arities. On the other hand, we show that the *max-closed* finite-valued cost functions of any fixed arity *cannot* express all finite-valued max-closed cost functions of any larger arity. Hence we identify an infinite hierarchy of finite-valued cost functions with ever-increasing expressive power.

Definition 26 *For all $d \geq 2$ we define the following:*

- $\mathbf{F}_{d,m}$ *denotes the set of all finite-valued cost functions (that is, cost functions whose valuation structure $\Omega = \mathbb{Q}_+$) of arity at most m over a domain of size d, and* $\mathbf{F}_d \stackrel{\text{def}}{=} \cup_{m \geq 0} \mathbf{F}_{d,m}$*;*
- $\mathbf{F}^{\max}_{d,m}$ *denotes the set of all finite-valued max-closed cost functions of arity at most m over an ordered domain of size d, and* $\mathbf{F}^{\max}_d \stackrel{\text{def}}{=} \cup_{m \geq 0} \mathbf{F}^{\max}_{d,m}$.

Cost functions from $\mathbf{F}_2$, that is, finite-valued cost functions over a Boolean domain, are also known as *pseudo-Boolean functions* [4]. The class of max-closed cost functions is discussed in more detail in [8] and shown to be tractable. A number of examples of max-closed cost functions are given in [8].

Proposition 27 *For all $d \geq 2$, $\langle \mathbf{F}_{d,1} \rangle \subsetneq \langle \mathbf{F}_{d,2} \rangle$ and $\langle \mathbf{F}^{\max}_{d,1} \rangle \subsetneq \langle \mathbf{F}^{\max}_{d,2} \rangle$.*

PROOF. Consider the binary weighted function $\mathcal{F} = \{\langle 1, \textsc{Min} \rangle, \langle 1, \textsc{Max} \rangle\}$. It is straightforward to verify that $\mathcal{F} \in \mathsf{fPol}(\mathbf{F}_{d,1})$ and $\mathcal{F} \in \mathsf{fPol}(\mathbf{F}^{\max}_{d,1})$.

Now consider the binary finite-valued max-closed cost function ϕ over any domain containing $\{0, 1\}$, defined by $\phi(\langle 0, 0 \rangle) = 1$ and $\phi(\langle ., . \rangle) = 0$ otherwise. Note that ϕ is max-closed but $\mathcal{F}$ is *not* a fractional polymorphism of ϕ. To see this, consider the tuples $\langle 0, 1 \rangle$ and $\langle 1, 0 \rangle$ (see the figure below).

$$\left.\begin{array}{rcc} 0\ 1 & \xrightarrow{\phi} & 0 \\ \underline{1\ 0} & & 0 \end{array}\right\} \sum = 0$$

$$\left.\begin{array}{rcc} \textsc{Min}\ \ 0\ 0 & \xrightarrow{\phi} & 1 \\ \textsc{Max}\ \ 1\ 1 & & 0 \end{array}\right\} \sum = 1$$

The result then follows from Theorem 10.

Now we prove a collapse result for the set of all finite-valued cost functions over an arbitrary finite domain. This result was previously known for the special case when $d = 2$: as we remarked earlier, any Boolean finite-valued cost function can be represented as a pseudo-Boolean function; using a well-known result from pseudo-Boolean optimisation [4], any such function can be expressed using quadratic pseudo-Boolean functions.

Theorem 28 *For all $d \geq 2$, $\langle \mathbf{F}_{d,2} \rangle = \mathbf{F}_d$.*

PROOF. As mentioned above, the case $d = 2$ follows from well-known results about pseudo-Boolean functions (see Theorem 1 of [4]). Let $\phi \in \mathbf{F}_{d,m}$ for some $d \geq 3$ and $m > 2$. We will show how to express ϕ using only unary and binary finite-valued cost functions. Without loss of generality, assume that all cost functions are defined over the set $D = \{0, 1, \ldots, M\}$, where $M = d - 1$, and denote by $D^m = \{t_1, \ldots, t_n\}$ the set of all m-tuples over D. Clearly, $n = d^m$. Let $K \in \mathbb{Q}_+$ be a fixed constant such that $K > \max_{t \in D^m} \phi(t)$. For any $e \in D$,

let χ^e be the binary finite-valued cost function defined by

$$\chi^e(x,y) = \begin{cases} 0 & \text{if } (x = e) \wedge (y = 0), \\ 0 & \text{if } (y \neq e) \wedge (y = 1), \\ K & \text{otherwise.} \end{cases}$$

For any $r \in \mathbb{Q}_+$, let μ^r be the unary finite-valued cost function defined by

$$\mu^r(z) = \begin{cases} r & \text{if } z = 0, \\ 0 & \text{otherwise.} \end{cases}$$

We now start building the gadget for ϕ. Let $x_1, \ldots, x_m$ be the variables upon which we wish to construct ϕ, and let $t_i \in D^m$ be an arbitrary fixed tuple. Figure 3 shows the part of the gadget for ϕ which ensures that the appropriate cost value is assigned to the tuple of values t_i. The complete gadget for ϕ consists of this part in n copies: one copy on a new set of variables for every $t_i \in D^m$.

Define new variables $y^i_1, \ldots, y^i_m$ and z^i. We apply cost functions on these variables as shown in Figure 3. Note that each variable y^i_j, $1 \leq j \leq m$, indi-

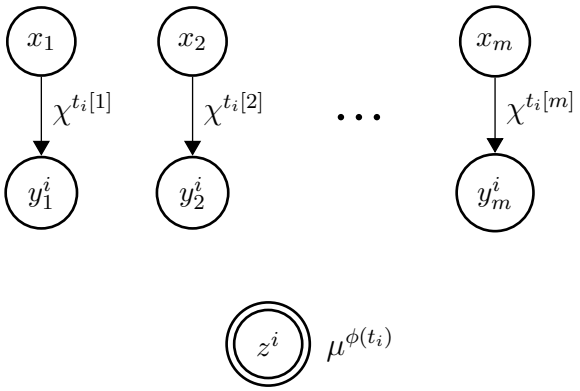

Figure 3. A part of the gadget for expressing ϕ in the proof of Theorem 28.

cates whether or not x_j is equal to $t_i[j]$: in any minimum-cost assignment, $(y^i_j = 0) \Leftrightarrow (x_j = t_i[j])$. It remains to define the constraints between the variables $y^i_1, \ldots, y^i_m$ and z^i. These will be chosen in such a way that any assignment of the values 0 or 1 to the variables y^i_j can be extended to an assignment to z^i with a total cost equal to the same fixed minimum value. Furthermore, in these extended assignments z^i is assigned 0 if and only if all the y^i_j are assigned 0. (We will achieve this by combining appropriate binary finite-valued cost functions over these variables and other fresh variables as described below.) Then, for every possible assignment of values t_i to the variables $x_1, \ldots, x_m$, there is exactly one z^i, $1 \leq i \leq n$, which is assigned the value 0 in any minimum-cost

extension of this assignment. The unary constraint with cost function $\mu^{\phi(t_i)}$ on each z^i then ensures that the complete gadget expresses ϕ.

To define the remaining constraints in Figure 3, we define two binary finite-valued cost functions as follows:

$$\phi_1(y,z) = \begin{cases} 0 & \text{if } (y=0) \wedge [(z=0) \vee (z=1)], \\ 0 & \text{if } (y \neq 0) \wedge (z \neq 0), \\ K & \text{otherwise} \end{cases}$$

and

$$\phi_2(y,z) = \begin{cases} 0 & \text{if } (y=0) \wedge [(z=0) \vee (z=2)], \\ 0 & \text{if } (y \neq 0) \wedge (z \neq 0), \\ K & \text{otherwise.} \end{cases}$$

Let $\mathcal{P} = \langle V, D, \mathcal{C} \rangle$ where $V = \{y_1, y_2, z\}$ and $\mathcal{C} = \{\langle\langle y_1, z\rangle, \phi_1\rangle, \langle\langle y_2, z\rangle, \phi_2\rangle\}$. (See Figure 4.)

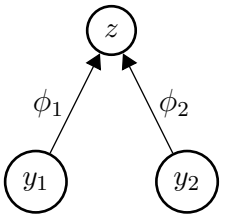

Figure 4. $\mathcal{P}$, an instance expressing or_2 over non-Boolean domains, from Theorem 28.

We define or_2 to be the cost function expressed by the gadget $\langle \mathcal{P}, \langle y_1, y_2, z\rangle\rangle$. The cost function $or_2(y_1, y_2, z)$ has the following properties:

- if both y_1, y_2 are assigned the zero value, then the total cost is 0 if and only if z is assigned the zero value, otherwise the total cost is either K (if $z = 1$ or $z = 2$) or $2K$ (if $z > 2$);
- if y_1 is assigned the zero value and y_2 a non-zero value, then the total cost is 0 if and only if z is assigned 1, otherwise the total cost is K;
- if y_1 is assigned a non-zero value and y_2 the zero value, then the total cost is 0 if and only if z is assigned 2, otherwise the total cost is K;
- if both x and y are assigned non-zero values, then the total cost is 0 if and only if z is assigned a non-zero value, otherwise the total cost is $2K$.

All these properties of or_2 can be easily verified by examining the so-called *microstructure* [24] of $\mathcal{P}$, as shown in Figure 5: this is a graph where the vertices are pairs $\langle v, e\rangle \in V \times D$, and two vertices $\langle v_1, e_1\rangle$ and $\langle v_2, e_2\rangle$ are connected by an edge with weight w if and only if there is a valued constraint $\langle\langle v_1, v_2\rangle, c\rangle \in \mathcal{C}$ such that $c(e_1, e_2) = w$.

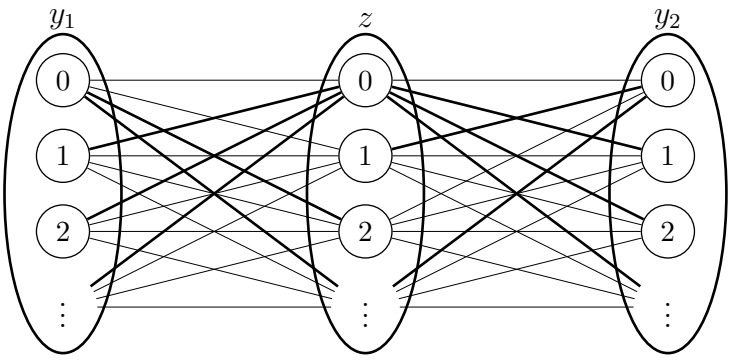

Figure 5. Microstructure of the instance $\mathcal{P}$ from Theorem 28: circles represent particular assignments to particular variables, as indicated, and edges are weighted by the cost of the corresponding pair of assignments. Thin edges indicate zero weight, and bold edges indicate weight K.

We have shown that, in any minimum-cost assignment for $\mathcal{P}$, the variable z takes the value 0 if and only if both of the variables y_1 and y_2 take the value 0. Hence the cost function or_2 can be viewed as a kind of 2-input "or-gate", with inputs y_1 and y_2 and output z. By cascading $m-1$ copies of this gadget we can express a cost function $or_m(y_1, \ldots, y_m, z)$, with the following properties:

- if the arguments $y_1, y_2, \ldots, y_m$ are all assigned the zero value, then assigning zero to z gives cost 0, but any non-zero assignment to z gives cost at least K;
- if not all the arguments $y_1, y_2, \ldots, y_m$ are assigned the zero value, then there is a non-zero value $e \in D$ such that assigning e to z gives cost 0, but assigning zero to z gives cost at least K.

Using this combined gadget on the variables $y_1^i, y_2^i, \ldots, y_m^i$ and z^i in Figure 3 completes the gadget for ϕ, and hence establishes that $\phi \in \langle \mathbf{F}_{d,2} \rangle$.

In contrast to this result, the remaining results in this section establish an infinite hierarchy of increasing expressive power for finite-valued *max-closed* cost functions. We will say that an m-tuple u *dominates* an m-tuple v, denoted $u \geq v$, if $u[i] \geq v[i]$ for $1 \leq i \leq m$.

Proposition 29 ([8]) *An m-ary cost function $\phi : D^m \to \Omega$ is max-closed if and only if* $\textsc{Max} \in \mathsf{FPol}(\{\phi\})$ *and ϕ is finitely antitone, that is, for all m-tuples u, v with $\phi(u), \phi(v) < \infty$, $u \leq v \Rightarrow \phi(u) \geq \phi(v)$.*

It follows that the finite-valued max-closed cost functions are simply the finite-valued antitone functions, that is, those functions whose values can only decrease as their arguments get larger. Note that for such functions the expressive power is likely to be rather limited because in any construction the "hidden variables" that are "projected out" can always be assigned the highest values

in their domain in order to minimise the cost. Hence, using such hidden variables only adds a constant value to the total cost, and so does not allow more cost functions to be expressed.

We now extend the separation result shown in Proposition 27 and separate each possible arity.

Proposition 30 *For all $d \geq 2$ and $m \geq 2$, $\{\langle m-1, \textsc{Max}_m\rangle, \langle 1, \textsc{Second}_m\rangle\} \in \mathsf{fPol}(\mathbf{F}^{\max}_{d,m-1})$.*

PROOF. Let ϕ be an arbitrary $(m-1)$-ary finite-valued max-closed cost function. Let $t_1, \ldots, t_m$ be $(m-1)$-tuples. We show that there is an i such that the tuple $s = \textsc{Second}_m(t_1, \ldots, t_m)$ dominates t_i, that is, $s[j] \geq t_i[j]$ for $1 \leq j \leq m-1$. To show this we count the number of tuples which can fail to be dominated by s. If a tuple t_p is not dominated by s, for some $1 \leq p \leq m$, it means that there is a position $1 \leq j \leq m-1$ such that $t_p[j] > s[j]$. But since $\textsc{Second}_m$ returns the second biggest value, for every $1 \leq j \leq m-1$, there is at most one tuple which is not dominated by s. Since there are $m \geq 3$ tuples, there must be an i such that t_i is dominated by s. Moreover, $\textsc{Max}_m(t_1, \ldots, t_m)$ clearly dominates all $t_1, \ldots, t_m$. By Proposition 29, ϕ is antitone and therefore $\{\langle m-1, \textsc{Max}_m\rangle, \langle 1, \textsc{Second}_m\rangle\}$ is a fractional polymorphism of ϕ, by Definition 9.

Proposition 31 *For all $d \geq 2$ and $m \geq 2$, $\{\langle m-1, \textsc{Max}_m\rangle, \langle 1, \textsc{Second}_m\rangle\} \notin \mathsf{fPol}(\mathbf{F}^{\max}_{d,m})$.*

PROOF. Let ϕ be the m-ary finite-valued max-closed cost function over any domain containing $\{0,1\}$, defined by $\phi(\langle 0, \ldots, 0\rangle) = 1$ and $\phi(\langle ., \ldots, .\rangle) = 0$ otherwise. To show that $\{\langle m-1, \textsc{Max}_m\rangle, \langle 1, \textsc{Second}_m\rangle\}$ is *not* a fractional polymorphism of ϕ, consider the m-tuples $\langle 0, \ldots, 0, 1\rangle$, $\langle 0, \ldots, 0, 1, 0\rangle, \ldots, \langle 1, 0, \ldots, 0\rangle$. Each of them is assigned cost 0 by ϕ. But applying the functions $\textsc{Max}_m$ ($(m-1)$ times) and $\textsc{Second}_m$ coordinate-wise results in $m-1$ tuples $\langle 1, \ldots, 1\rangle$, which are assigned cost 0 by ϕ, and one tuple $\langle 0, \ldots, 0\rangle$, which is assigned cost 1 by ϕ (see Figure 6).

Theorem 32 *For all $d \geq 2$, $\langle \mathbf{F}^{\max}_{d,1}\rangle \subsetneq \langle \mathbf{F}^{\max}_{d,2}\rangle \subsetneq \langle \mathbf{F}^{\max}_{d,3}\rangle \subsetneq \langle \mathbf{F}^{\max}_{d,4}\rangle \cdots$*

PROOF. By Propositions 30 and 31 and Theorem 10.

Figure 7 summarises the results from this section.

$$
\begin{array}{rl}
 & \left.\begin{array}{l} 0\,0\,\ldots\,0\,0\,1 \\ 0\,0\,\ldots\,0\,1\,0 \\ \vdots \\ 1\,0\,\ldots\,0\,0\,0 \end{array}\right. \\
\hline
\textsc{Max}_m & 1\,1\,\ldots\,1\,1\,1 \\
\vdots & \vdots \\
\textsc{Max}_m & 1\,1\,\ldots\,1\,1\,1 \\
\textsc{Second}_m & 0\,0\,\ldots\,0\,0\,0
\end{array}
\quad
\begin{array}{c}
\left.\begin{array}{c} 0 \\ 0 \\ \vdots \\ 0 \end{array}\right\} \sum = 0 \\
\hline
\left.\begin{array}{c} 0 \\ \vdots \\ 0 \\ 1 \end{array}\right\} \sum = 1
\end{array}
$$

Figure 6. $\{\langle m-1, \textsc{Max}_m\rangle, \langle 1, \textsc{Second}_m\rangle\} \notin \mathsf{fPol}(\{\phi\})$ for ϕ from Proposition 31.

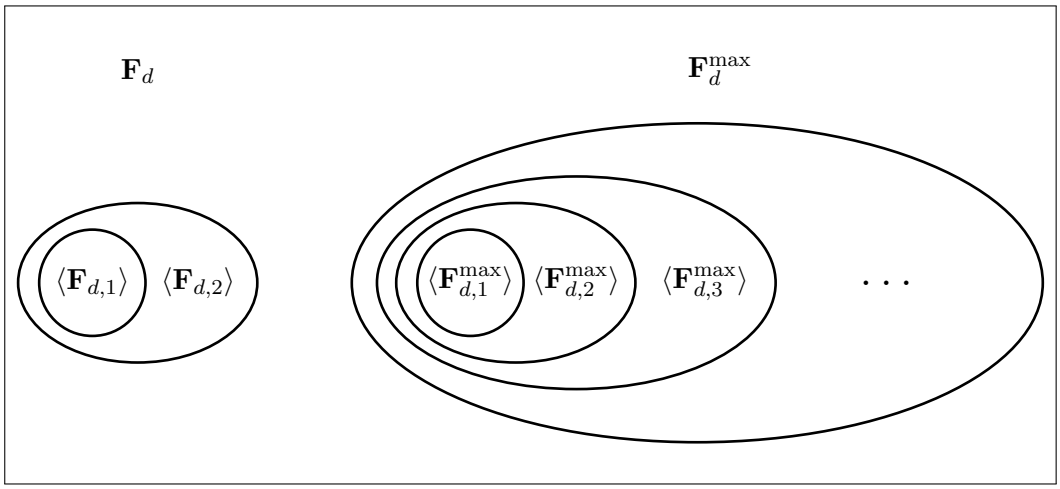

Figure 7. Summary of results from Section 5, for all $d \geq 2$.

6 General Cost Functions

In this section, we show that general cost functions of a fixed arity can express cost functions of arbitrary arities. Comparing this result with the results of the previous section provides a striking example of the way in which allowing infinite cost values in a valued constraint language can drastically affect the expressibility of cost functions over that language, including finite-valued cost functions.

Definition 33 *For all $d \geq 2$ we define the following:*

- $\mathbf{G}_{d,m}$ *denotes the set of all general cost functions (that is, cost functions whose valuation structure $\Omega = \overline{\mathbb{Q}}_+$) of arity at most m over a domain of size d, and $\mathbf{G}_d \stackrel{\text{def}}{=} \cup_{m \geq 0} \mathbf{G}_{d,m}$;*
- $\mathbf{G}_{d,m}^{\max}$ *denotes the set of all general max-closed cost functions of arity at*

most m over an ordered domain of size d, and $\mathbf{G}_d^{\max} \stackrel{\text{def}}{=} \cup_{m\geq 0}\mathbf{G}_{d,m}^{\max}$.

The class of general max-closed cost functions is known to be tractable [8].

Once again it is straightforward to establish a separation between unary and binary general cost functions.

Proposition 34 *$\langle\mathbf{G}_{d,1}\rangle \subsetneq \langle\mathbf{G}_{d,2}\rangle$ and $\langle\mathbf{G}_{d,1}^{\max}\rangle \subsetneq \langle\mathbf{G}_{d,2}^{\max}\rangle$.*

PROOF. Identical to the proof of Proposition 27.

As with crisp cost functions, in the special case of a Boolean domain, we can show a separation between binary and ternary general cost functions.

Proposition 35 *$\langle\mathbf{G}_{2,2}\rangle \subsetneq \langle\mathbf{G}_{2,3}\rangle$ and $\langle\mathbf{G}_{2,2}^{\max}\rangle \subsetneq \langle\mathbf{G}_{2,3}^{\max}\rangle$.*

PROOF. By Proposition 19, Majority $\in \mathsf{FPol}(\mathbf{G}_{2,2})$ and Majority $\in \mathsf{FPol}(\mathbf{G}_{2,2}^{\max})$. By Proposition 20, Majority $\notin \mathsf{FPol}(\mathbf{G}_{2,3})$ and Majority $\notin \mathsf{FPol}(\mathbf{G}_{2,3}^{\max})$. The result then follows by Theorem 10.

Next we show a collapse result for general cost functions.

Theorem 36 *For all $d \geq 3$, $\langle\mathbf{G}_{d,1}\rangle \subsetneq \langle\mathbf{G}_{d,2}\rangle = \mathbf{G}_d$. Moreover, $\langle\mathbf{G}_{2,1}\rangle \subsetneq \langle\mathbf{G}_{2,2}\rangle \subsetneq \langle\mathbf{G}_{2,3}\rangle = \mathbf{G}_2$.*

PROOF. Let $\phi \in \mathbf{G}_{d,m}$ for some $d \geq 3$ and $m > 2$. It is easy to check that the same construction as in the proof of Theorem 28 can be used to express ϕ, with $K = \infty$.

Now let $\phi \in \mathbf{G}_{2,m}$ for some $m > 2$. It is easy to check that a similar construction to that used in the proof of Theorem 28 can be used to express ϕ, where the instance $\mathcal{P}$ is replaced by the ternary Boolean relation which expresses the truth table of a 2-input or-gate.

Note that the proof shows a slightly stronger result: $\mathbf{G}_2 = \langle\mathbf{R}_{2,3} \cup \mathbf{F}_{2,1}\rangle$, and for all $d \geq 3$, $\mathbf{G}_d = \langle\mathbf{R}_{d,2} \cup \mathbf{F}_{d,1}\rangle$. In other words, all general cost functions can be expressed using *unary* finite-valued cost functions together with ternary relations (in the case $d = 2$), or binary relations (in the case $d \geq 3$).

By investigating feasibility polymorphisms and fractional polymorphisms, we will now show a collapse result for general max-closed cost functions.

First we show that general max-closed cost functions of a fixed arity have the same feasibility polymorphisms as max-closed cost functions of arbitrary arities.

Proposition 37 *For all* $d \geq 3$, $\mathsf{FPol}(\mathbf{G}^{\max}_{d,2}) = \mathsf{FPol}(\mathbf{G}^{\max}_{d})$. *Moreover,* $\mathsf{FPol}(\mathbf{G}^{\max}_{2,3}) = \mathsf{FPol}(\mathbf{G}^{\max}_{2})$.

PROOF. Assume for contradiction, that there is an $f \in \mathsf{FPol}(\mathbf{G}^{\max}_{d,2})$, such that $f \notin \mathsf{FPol}(\mathbf{G}^{\max}_{d})$. By Definition 33, $\{\mathsf{Feas}(\phi) \mid \phi \in \mathbf{G}^{\max}_{d}\} = \mathbf{R}^{\max}_{d}$. Therefore, such an f would contradict Theorem 25 since $\mathsf{Pol}(\mathbf{R}^{\max}_{d,2}) = \mathsf{Pol}(\mathbf{R}^{\max}_{d})$.

Similarly, assume that there is an $f \in \mathsf{FPol}(\mathbf{G}^{\max}_{2,3})$ such that $f \notin \mathsf{FPol}(\mathbf{G}^{\max}_{2})$. This would contradict Theorem 22 since $\mathsf{Pol}(\mathbf{R}^{\max}_{2,3}) = \mathsf{Pol}(\mathbf{R}^{\max}_{2})$.

We now prove that general max-closed cost functions of a fixed arity have the same fractional polymorphisms as general max-closed cost functions of arbitrary arities. First we characterise the feasibility polymorphisms of general max-closed cost functions.

Proposition 38 *For all* $d \geq 2$,

$$\mathsf{FPol}(\mathbf{G}^{\max}_{d}) = \{\textsc{Max}_I \mid \emptyset \neq I \subseteq \{1, \ldots, k\}, k = 1, 2, \ldots\}.$$

PROOF. It follows from Definition 33 that $\{\mathsf{Feas}(\phi) \mid \phi \in \mathbf{G}^{\max}_{d}\} = \mathbf{R}^{\max}_{d}$. Therefore, $\mathsf{FPol}(\mathbf{G}^{\max}_{d}) = \mathsf{FPol}(\mathbf{R}^{\max}_{d})$ and the result follows from Proposition 17.

Next we characterise the fractional polymorphisms of general max-closed cost functions.

Definition 39 *Let* $\mathcal{F} = \{(r_1, \textsc{Max}_{S_1}), \ldots, (r_n, \textsc{Max}_{S_n})\}$ *be a* k*-ary weighted function and* $S \subseteq \{1, \ldots, k\}$.

We define

$$\mathrm{supp}_{\mathcal{F}} S \stackrel{\text{def}}{=} \{i \mid S_i \cap S \neq \emptyset\},$$

and

$$\mathrm{wt}_{\mathcal{F}}(S) \stackrel{\text{def}}{=} \sum_{i \in \mathrm{supp}_{\mathcal{F}}(S)} r_i.$$

Theorem 40 *Let* $\mathcal{F} = \{(r_1, \textsc{Max}_{S_1}), \ldots, (r_n, \textsc{Max}_{S_n})\}$ *be a* k*-ary weighted function. The following are equivalent:*

(1) $\mathcal{F} \in \mathsf{fPol}(\mathbf{G}_d^{\max})$.
(2) $\mathcal{F} \in \mathsf{fPol}(\mathbf{G}_{d,1}^{\max})$.
(3) For every subset $S \subseteq \{1, \ldots, k\}$, $\mathrm{wt}_{\mathcal{F}}(S) \geq |S|$.

PROOF.

We first show that $\neg(3) \Rightarrow \neg(2) \Rightarrow \neg(1)$.

First suppose that there exists an $S \subseteq \{1, \ldots, k\}$ such that $\mathrm{wt}_{\mathcal{F}}(S) < |S|$. Let $\{a, b\} \subseteq D$ be the two biggest elements of D and $a < b$. Consider the unary cost function ϕ where

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

Certainly $\phi \in \mathbf{G}_{d,1}^{\max}$.

Now let

$$x_i = \begin{cases} b & \text{if } i \in S, \\ a & \text{if } i \notin S. \end{cases}$$

We have that

$$\begin{aligned} \sum_{i=1}^{k} \phi(x_i) &= k - |S|, \text{ and} \\ \sum_{j=1}^{n} r_j \phi(\textsc{Max}_{S_j}(x_1, \ldots, x_k)) &= \sum_{j \notin \mathrm{supp}_{\mathcal{F}} S} r_j \phi(a) + \sum_{j \in \mathrm{supp}_{\mathcal{F}} S} r_j \phi(b) \\ &= k - \mathrm{wt}_{\mathcal{F}}(S) \\ &> k - |S|, \text{ by assumption.} \end{aligned}$$

So $\mathcal{F}$ is not a fractional polymorphism of $\mathbf{G}_{d,1}^{\max}$, and hence not a fractional polymorphism of $\mathbf{G}_d^{\max}$.

To complete the proof we will show that (3) $\Rightarrow$ (1).

Suppose that, for every subset $S \subseteq \{1, \ldots, k\}$, $\mathrm{wt}_{\mathcal{F}}(S) \geq |S|$.

We will first show the existence of a set of non-negative values p_{ji} for $j = 1, \ldots, n$ and $i = 1, \ldots, k$ where

$$\begin{aligned} \sum_{i=1}^{k} p_{ji} &= r_j, \\ \sum_{j=1}^{n} p_{ji} &= 1 \quad \text{and} \\ p_{ji} &= 0 \text{ if } i \notin S_j. \end{aligned}$$

Consider the network in Figure 8. The capacity from the source to any node x_i is one. The capacity from node y_j to the sink is r_j. There is an arc from node x_i to node y_j precisely when $i \in S_j$, and the capacity of these arcs is infinite.

We will use the MIN-CUT MAX-FLOW theorem to generate the p_{ji}.

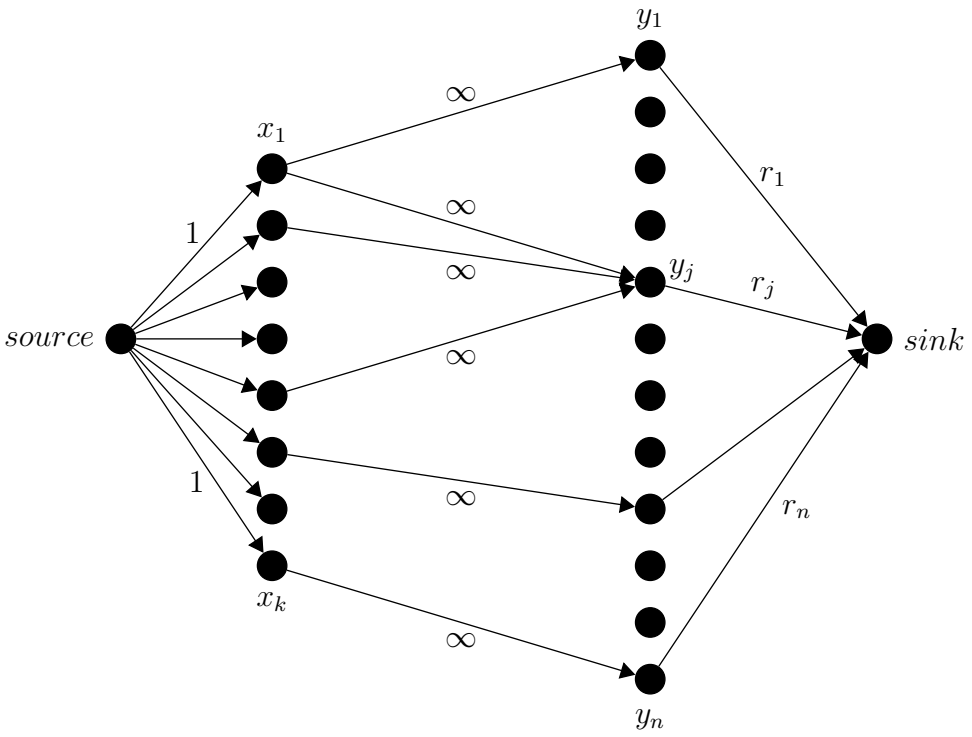

Figure 8. The flow from x_i to y_j in a maximum flow is the value of p_{ji}

Suppose that we have a minimum cut of this network. Let A be those arcs in this cut from the source to any node x_i. Let $S = \{1, \ldots, k\} - \{i \mid x_i \in A\}$. Since we have a cut we must (at least) cut every arc from the nodes $\{y_j \mid j \in \text{supp}_{\mathcal{F}}(S)\}$ to the sink. By assumption $\text{wt}_{\mathcal{F}}(S) \geq |S|$ and so this cut has total cost at least k. Certainly there is a cut of cost exactly k (cut all arcs from the source), and so the max-flow through this network is precisely k. Such a flow can only be achieved if each arc from the source and each arc to the sink

is filled to its capacity. The flow along the arc from x_i to y_j then gives the required value for p_{ji}.

Now we will use these values p_{ji} to show that $\mathcal{F}$ is indeed a fractional polymorphism of $\mathbf{G}_d^{\max}$.

Let $t_1, \ldots, t_k$ be m-ary tuples and $\phi \in \mathbf{G}_{d,m}^{\max}$ be an m-ary cost function. We have to show the following:

$$\sum_{i=1}^{k} \phi(t_i) \geq \sum_{j=1}^{n} r_j \phi(\textsc{Max}_{S_j}(t_1, \ldots, t_k)). \tag{1}$$

If any $\phi(t_i)$ is infinite, then this inequality clearly holds.

By Proposition 38, all $\textsc{Max}_{S_j}$, $1 \leq j \leq n$, are feasibility polymorphisms of $\mathbf{G}_d^{\max}$. Therefore, if all $\phi(t_i)$ are finite, then all $\phi(\textsc{Max}_{S_j}(t_1, \ldots, t_k))$ are finite as well.

By definition of p_{ji} and using that $p_{ji} = 0$ whenever $i \notin S_j$ we have that

$$\sum_{j=1}^{n} r_j \phi(\textsc{Max}_{S_j}(t_1, \ldots, t_k)) = \sum_{j=1}^{n} \sum_{i \in S_j} p_{ji} \phi(\textsc{Max}_{S_j}(t_1, \ldots, t_k)).$$

Now, since ϕ is antitone, we have

$$\sum_{j=1}^{n} \sum_{i \in S_j} p_{ji} \phi(\textsc{Max}_{S_j}(t_1, \ldots, t_k)) \leq \sum_{j=1}^{n} \sum_{i \in S_j} p_{ji} \phi(t_i)$$

Since $p_{ji} = 0$ whenever $i \notin S_j$ we have that

$$\sum_{j=1}^{n} \sum_{i \in S_j} p_{ji} \phi(t_i) = \sum_{j=1}^{n} \sum_{i=1}^{k} p_{ji} \phi(t_i)$$

Finally, since $\sum_{j=1}^{n} p_{ji} = 1$ we have established Inequality (1).

Theorem 41 *For all $d \geq 3$, $\mathsf{fPol}(\mathbf{G}_{d,2}^{\max}) = \mathsf{fPol}(\mathbf{G}_d^{\max})$. Moreover, $\mathsf{fPol}(\mathbf{G}_{2,3}^{\max}) = \mathsf{fPol}(\mathbf{G}_2^{\max})$.*

PROOF. By Proposition 37, $\mathbf{G}_{d,2}^{\max}$ and $\mathbf{G}_d^{\max}$ have the same feasibility polymorphisms. Also, $\mathbf{G}_{2,3}^{\max}$ and $\mathbf{G}_2^{\max}$ have the same feasibility polymorphisms.

By Proposition 38, these feasibility polymorphisms are of the form "max-on-a-subset". Clearly, each component function of a fractional polymorphism has to be a feasibility polymorphism. Therefore, the result follows from Theorem 40.

Theorem 42 *For all $d \geq 3$, $\langle \mathbf{G}_{d,1}^{\max} \rangle \subsetneq \langle \mathbf{G}_{d,2}^{\max} \rangle = \mathbf{G}_d^{\max}$. Moreover, $\langle \mathbf{G}_{2,1}^{\max} \rangle \subsetneq \langle \mathbf{G}_{2,2}^{\max} \rangle \subsetneq \langle \mathbf{G}_{2,3}^{\max} \rangle = \mathbf{G}_2^{\max}$.*

PROOF.

The separation results were obtained in Propositions 34 and 35, by showing that the valued constraint languages involved have different feasibility polymorphisms.

For all $d' \geq 2$, $m \geq 1$ and $c \in \mathbb{Q}_+$, $\mathbf{G}_{d',m}^{\max}$ is closed under scaling by c. Therefore, using Theorem 10, the collapses follow from Proposition 37 and Theorem 41.

Note that the proof shows a slightly stronger result: for all $d \geq 3$, $\mathbf{G}_d^{\max} = \langle \mathbf{R}_{d,2}^{\max} \cup \mathbf{F}_{d,1}^{\max} \rangle$, and $\mathbf{G}_2^{\max} = \langle \mathbf{R}_{2,3}^{\max} \cup \mathbf{F}_{2,1}^{\max} \rangle$.

A constructive, gadget-based proof of Theorem 42 can be found in [33].

Figure 9 summarises the results from this section.

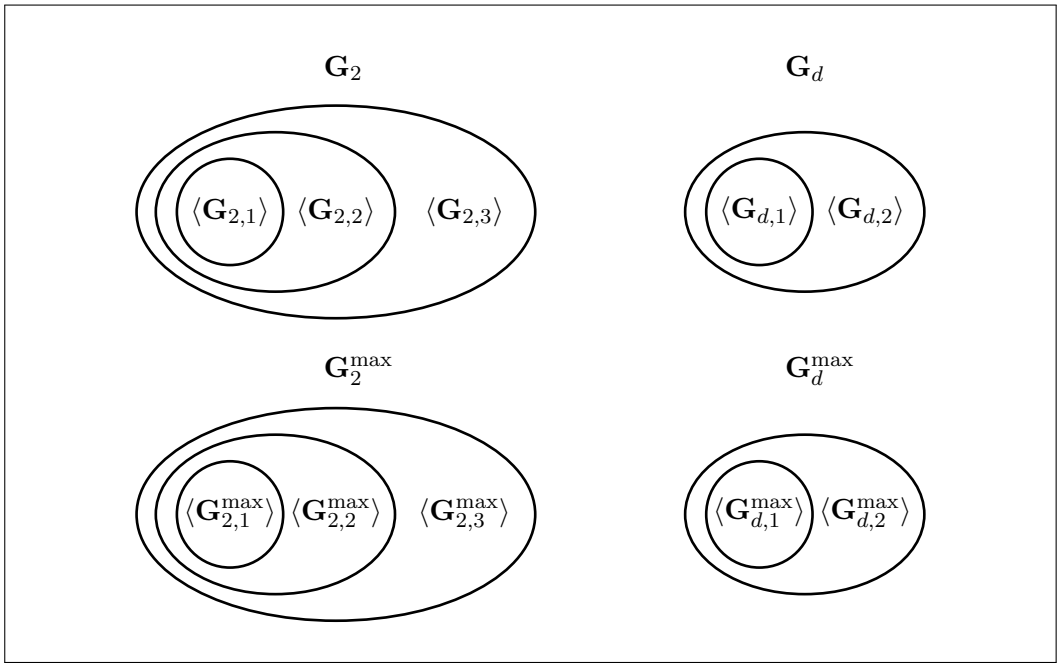

Figure 9. Summary of results from Section 6, for all $d \geq 3$.

Example 43 *Consider the ternary finite-valued max-closed cost function ϕ over $D = \{0, 1, 2\}$ which is defined by*

$$\phi(t) = \begin{cases} 1 & \text{if } t = \langle 0, 0, 0 \rangle, \\ 0 & \text{otherwise.} \end{cases}$$

By Proposition 31, $\phi \notin \langle \mathbf{F}_{3,2}^{\max} \rangle$. In other words, ϕ is not expressible using only finite-valued max-closed cost functions of arity at most 2. However, by Theorem 42, $\phi \in \langle \mathbf{G}_{3,2}^{\max} \rangle$. We now show how ϕ can be expressed using general max-closed cost functions of arity at most 2.

Let ϕ_0 be the binary finite-valued max-closed cost function defined as follows:

$$\phi_0(t) = \begin{cases} 1 & \text{if } t = \langle 0, 0 \rangle, \\ 0 & \text{otherwise.} \end{cases}$$

Next, define two binary crisp[4] *max-closed cost functions*

$$\phi_1(t) = \begin{cases} \infty & \text{if } t = \langle 0, 1 \rangle, \\ 0 & \text{otherwise} \end{cases}$$

and

$$\phi_2(t) = \begin{cases} \infty & \text{if } t = \langle 0, 2 \rangle, \\ 0 & \text{otherwise.} \end{cases}$$

Let $\mathcal{P} = \langle V, D, \mathcal{C} \rangle$ where $V = \{x, y, z, u, v\}$ and

$$\mathcal{C} = \{\langle \langle x, u \rangle, \phi_1 \rangle, \langle \langle y, u \rangle, \phi_2 \rangle, \langle \langle y, v \rangle, \phi_1 \rangle, \langle \langle z, v \rangle, \phi_2 \rangle, \langle \langle u, v \rangle, \phi_0 \rangle\}.$$

We claim that $\langle \mathcal{P}, \langle x, y, z \rangle \rangle$ is a gadget for expressing ϕ over $\mathbf{G}_{3,2}^{\max}$. (See Figure 10.) If any of x, y, z is non-zero, then at least one of the variables u, v can be assigned a non-zero value and the cost of such an assignment is 0. Conversely, if x, y and z are all assigned zero, then the minimum-cost assignment must also assign zero to both u and v, and hence has cost 1.

We now show another gadget for expressing ϕ, using only crisp max-closed cost functions of arity at most 2 and finite-valued max-closed cost functions of arity at most 1.

[4] Note that a "finite variant" of ϕ_1, defined as $\phi_1(\langle 0, 1 \rangle) = K$ for some finite $K < \infty$ and $\phi_1(\langle ., . \rangle) = 0$ otherwise, is not max-closed. The infinite cost is necessary.

Let μ be the unary finite-valued max-closed cost function defined by

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

Let $\mathcal{P}' = \langle V', D, \mathcal{C}' \rangle$ where $V' = \{x, y, z, u, v, w\}$ and

$$\mathcal{C} = \{\langle\langle x, u\rangle, \phi_1\rangle, \langle\langle y, u\rangle, \phi_2\rangle, \langle\langle y, v\rangle, \phi_1\rangle, \langle\langle z, v\rangle, \phi_2\rangle, \langle\langle u, w\rangle, \phi_1\rangle, \langle\langle v, w\rangle, \phi_2\rangle, \langle w, \mu\rangle\}.$$

See Figure 11. Similarly to the argument above, $\langle \mathcal{P}', \langle x, y, z\rangle\rangle$ is a gadget for expressing ϕ. This can be verified by examining the microstructure of $\mathcal{P}'$ (see Figure 12).

7 Conclusions and Open Problems

We have investigated the expressive power of valued constraints in general and max-closed valued constraints in particular.

In the case of relations, we built on previously known results about the expressibility of an arbitrary relation in terms of binary or ternary relations. We were able to prove, in a similar way, that an arbitrary max-closed relation can be expressed using binary or ternary max-closed relations. The results about the collapse of the set of all relations and all max-closed relations contrast sharply with the case of finite-valued cost functions, where we showed an infinite hierarchy for max-closed cost functions. This shows that the VCSP is not just a minor generalisation of the CSP – finite-valued max-closed cost functions behave very differently from crisp max-closed cost functions with respect to expressive power. We also showed the collapse of general cost functions, by characterising the feasibility polymorphisms and fractional polymorphisms of general max-closed cost functions. This shows that allowing infinite costs in max-closed cost functions increases their expressive power substantially, and sometimes allows more finite-valued functions to be expressed.

We remark that all of our results about max-closed cost functions obviously have equivalent versions for *min-closed* cost functions, that is, those which have the fractional polymorphism $\{\langle 2, \text{MIN}\rangle\}$. In the Boolean crisp case these are precisely the relations that can be expressed by a conjunction of **Horn** clauses.

One of the reasons why understanding the expressive power of valued con-

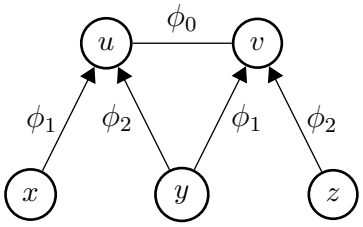

Figure 10. $\mathcal{P}$, an instance of $\mathsf{VCSP}(\mathbf{G}^{\max}_{3,2})$ expressing ϕ, from Example 43.

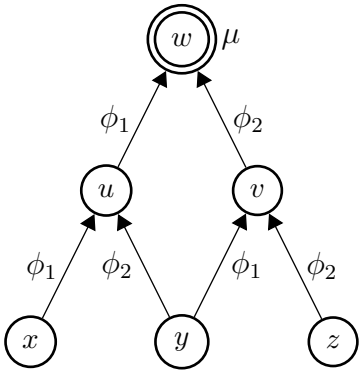

Figure 11. $\mathcal{P}'$, an instance of $\mathsf{VCSP}(\mathbf{R}^{\max}_{3,2} \cup \mathbf{F}^{\max}_{3,1})$ expressing ϕ, from Example 43.

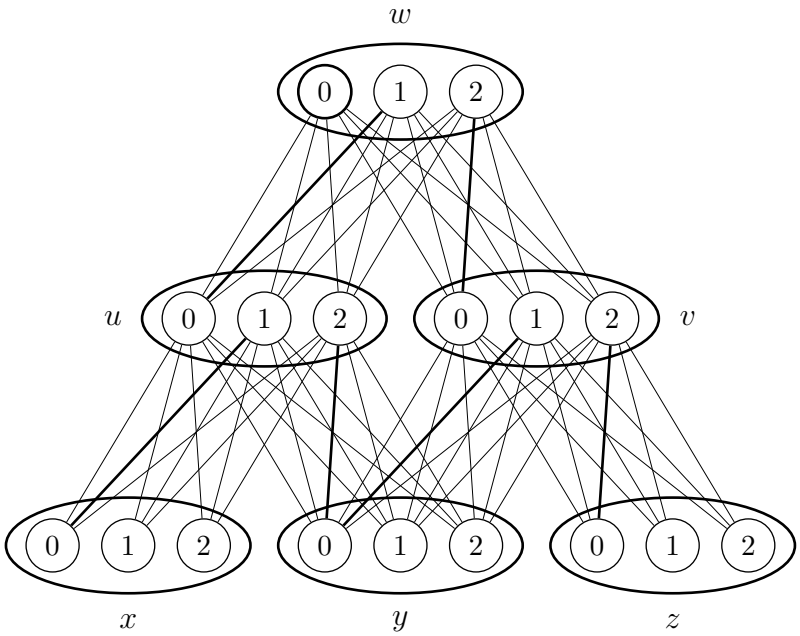

Figure 12. Microstructure of the instance $\mathcal{P}'$ from Example 43: circles represent particular assignments to particular variables, as indicated, and edges are weighted by the cost of the corresponding pair of assignments. Thin edges indicate zero weight, bold edges indicate infinite weight, and assigning 0 to variable w gives cost 1.

straints is important, is for the investigation of **submodular functions**. A cost function ϕ is called submodular if it has the fractional polymorphism $\{\langle 1, \textsc{Min}\rangle, \langle 1, \textsc{Max}\rangle\}$. The standard problem of submodular function minimisation corresponds to solving a VCSP with submodular cost functions over the Boolean domain [7].

Submodular function minimisation (SFM) is a central problem in discrete optimisation, with links to many different areas [14,27,32,17]. Although it has been known for a long time that the ellipsoid algorithm can be used to solve SFM in polynomial time, this algorithm is not efficient in practice. Relatively recently, several new strongly polynomial combinatorial algorithms have been discovered for SFM [31,14,16,18]. Unfortunately, the time complexity of the fastest published algorithm for SFM is roughly of an order of $O(n^6)$, where n is the total number of variables [28].

However, for certain special cases of SFM, more efficient algorithms are known to exist. For example, the (weighted) Min-Cut problem is a special case of SFM that can be solved in cubic time [14]. Moreover, it is known that SFM over a Boolean domain can be solved in $O(n^3)$ time, when the submodular function f satisfies various extra conditions [1,9,29,34]. In particular, in the case of non-Boolean domains, a cubic-time algorithm exists for SFM when f can be expressed as a sum of *binary* submodular functions [7].

These observations naturally raise the following question: what is the most general class of submodular functions that can be minimised in cubic time (or better)? One way to tackle this question is to investigate the expressive power of particular submodular functions which are known to be solvable in cubic time. Any fixed set of functions which can be *expressed* using such functions can be reduced to a cubic time problem, by replacing certain constraints with gadgets [6].

One intriguing result is already known for submodular *relations*. In the case of relations, having $\{\langle 1, \textsc{Min}\rangle, \langle 1, \textsc{Max}\rangle\}$ as a fractional polymorphism implies having both Min and Max as polymorphisms. The ternary Median operation can be obtained by composing the operations Min and Max, so all submodular relations have the Median operation as a polymorphism. It follows that submodular relations are binary decomposable [20], and hence all submodular relations are expressible using binary submodular relations over the same variables.

For finite-valued and general submodular cost functions it is an important open question as to whether they can be expressed using submodular cost functions of some fixed arity. If they can, then this raises the possibility of designing new, more efficient, algorithms for submodular function minimisation.

Acknowledgements

The authors would like to thank Martin Cooper, Martin Green, Chris Jefferson, Karen Petrie and András Salamon for many helpful discussions, and the anonymous reviewers for useful comments on an earlier draft of this paper.

References

[1] A. Billionet, M. Minoux, Maximizing a supermodular pseudo-boolean function: a polynomial algorithm for cubic functions, Discrete Applied Mathematics 12 (1985) 1–11.

[2] S. Bistarelli, H. Fargier, U. Montanari, F. Rossi, T. Schiex, G. Verfaillie, Semiring-based CSPs and valued CSPs: Frameworks, properties, and comparison, Constraints 4 (1999) 199–240.

[3] E. Boehler, S. Reith, H. Schnoor, H. Vollmer, Bases for Boolean co-clones, Information Processing Letters 96 (2005) 59–66.

[4] E. Boros, P. L. Hammer, Pseudo-boolean optimization, Discrete Applied Mathematics 123 (1-3) (2002) 155–225.

[5] A. Bulatov, A. Krokhin, P. Jeavons, Classifying the complexity of constraints using finite algebras, SIAM Journal on Computing 34 (3) (2005) 720–742.

[6] D. Cohen, M. Cooper, P. Jeavons, An algebraic characterisation of complexity for valued constraints, in: CP'06, vol. 4204 of LNCS, 2006.

[7] D. Cohen, M. Cooper, P. Jeavons, A. Krokhin, A maximal tractable class of soft constraints, Journal of Artificial Intelligence Research (JAIR) 22 (2004) 1–22.

[8] D. Cohen, M. Cooper, P. Jeavons, A. Krokhin, The complexity of soft constraint satisfaction, Artificial Intelligence 170 (2006) 983–1016.

[9] 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.

[10] R. Dechter, On the expressiveness of networks with hidden variables, in: AAAI, 1990.

[11] R. Dechter, J. Pearl, Tree clustering for constraint networks, Artificial Intelligence 38 (3) (1989) 353–366.

[12] K. Denecke, S. Wismath, Universal Algebra and Applications in Theoretical Computer Science, Chapman and Hall/CRC Press, 2002.

[13] A. Fearnley, A strongly rigid binary relation, Acta Sci. Math. (Szeged) 61 (1995) 35–41.

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

[15] M. Garey, D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco, CA., 1979.

[16] S. Iwata, A faster scaling algorithm for minimizing submodular functions, SIAM Journal on Computing 32 (2003) 833–840.

[17] S. Iwata, Submodular function minimization, Mathematical Programming 112 (2008) 45–64.

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

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

[20] P. Jeavons, D. Cohen, M. Cooper, Constraints, consistency and closure, Artificial Intelligence 101 (1–2) (1998) 251–265.

[21] P. Jeavons, D. Cohen, M. Gyssens, Closure properties of constraints, Journal of the ACM 44 (1997) 527–548.

[22] P. Jeavons, D. Cohen, M. Gyssens, How to determine the expressive power of constraints, Constraints 4 (1999) 113–131.

[23] P. Jeavons, M. Cooper, Tractable constraints on ordered domains, Artificial Intelligence 79 (2) (1995) 327–339.

[24] P. Jégou, Decomposition of domains based on the micro-structure of finite constraint-satisfaction problems, in: AAAI, 1993.

[25] J. Larrosa, R. Dechter, On the dual representation of non-binary semiring-based CSPs, in: Workshop on Soft Constraints – CP'00, 2000.

[26] U. Montanari, Networks of constraints: Fundamental properties and applications to picture processing, Information Sciences 7 (1974) 95–132.

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

[28] J. B. Orlin, A faster strongly polynomial time algorithm for submodular function minimization., in: IPCO'07, vol. 4513 of LNCS, 2007.

[29] M. Queyranne, Minimising symmetric submodular functions, Mathematical Programming 82 (1998) 3–12.

[30] F. Rossi, P. van Beek, T. Walsh (eds.), The Handbook of Constraint Programming, Elsevier, 2006.

[31] A. Schrijver, A combinatorial algorithm minimizing submodular functions in strongly polynomial time, Journal of Combinatorial Theory, Series B 80 (2000) 346–355.

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

[33] B. Zanuttini, S. Živný, A note on some collapse results of valued constraints, submitted for publication (2008).

[34] S. Živný, P. Jeavons, Classes of submodular constraints expressible by graph cuts, in: CP'08, vol. 5202 of LNCS, 2008.