

THE POWER OF LINEAR PROGRAMMING FOR GENERAL-VALUED CSPS*

VLADIMIR KOLMOGOROV[†], JOHAN THAPPER[‡], AND STANISLAV ŽIVNÝ[§]

Abstract. Let D , called the domain, be a fixed finite set and let Γ , called the valued constraint language, be a fixed set of functions of the form $f : D^m \rightarrow \mathbb{Q} \cup \{\infty\}$, where different functions might have different arity m . We study the *valued constraint satisfaction problem* parametrised by Γ , denoted by $\text{VCSP}(\Gamma)$. These are minimisation problems given by n variables and the objective function given by a sum of functions from Γ , each depending on a subset of the n variables. For example, if $D = \{0, 1\}$ and Γ contains all ternary $\{0, \infty\}$ -valued functions, $\text{VCSP}(\Gamma)$ corresponds to 3-SAT. More generally, if Γ contains only $\{0, \infty\}$ -valued functions, $\text{VCSP}(\Gamma)$ corresponds to $\text{CSP}(\Gamma)$. If $D = \{0, 1\}$ and Γ contains all ternary $\{0, 1\}$ -valued functions, $\text{VCSP}(\Gamma)$ corresponds to Min-3-SAT, in which the goal is to minimise the number of unsatisfied clauses in a 3-CNF instance. *Finite-valued* constraint languages contain functions that take on only rational values and not infinite values.

Our main result is a precise algebraic characterisation of valued constraint languages whose instances can be solved *exactly* by the *basic linear programming relaxation* (BLP). For a valued constraint language Γ , BLP is a decision procedure for Γ if and only if Γ admits a symmetric fractional polymorphism of *every* arity. For a finite-valued constraint language Γ , BLP is a decision procedure if and only if Γ admits a symmetric fractional polymorphism of *some* arity, or equivalently, if Γ admits a symmetric fractional polymorphism of *arity 2*.

Using these results, we obtain tractability of several novel classes of problems, including problems over valued constraint languages that are: (1) submodular on *arbitrary lattices*; (2) k -submodular on *arbitrary finite domains*; (3) weakly (and hence strongly) tree-submodular on *arbitrary trees*.

Key words. valued constraint satisfaction, fractional polymorphisms, submodularity, bisubmodularity, linear programming

AMS subject classifications. 08A70, 68Q25, 68Q17, 90C27

1. Introduction.

1.1. Constraint Satisfaction. The constraint satisfaction problem provides a common framework for many theoretical and practical problems in computer science [31]. An instance of the *constraint satisfaction problem* (CSP) consists of a collection of variables that must be assigned labels from a given domain subject to specified constraints [60]. The CSP is equivalent to the problem of evaluating conjunctive queries on databases [47], and to the homomorphism problem for relational structures [25].

The classic 3-COLOUR problem can be seen as the following CSP: the domain consists of three labels corresponding to the three colours; the variables correspond to the vertices of the graph; and the constraints specify that the variables corresponding to adjacent vertices have to be assigned different labels.

The CSP is NP-complete in general and thus one is interested in restrictions which give rise to tractable classes of problems. One possibility is to restrict the structure

*Part of this work (by J. Thapper and S. Živný) appeared in the *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 669–678, 2012 [69]. Part of this work (by V. Kolmogorov) appeared in the *Proceedings of the 40th International Colloquium on Automata, Languages and Programming (ICALP)*, pp. 625–636, 2013 [49]. Vladimir Kolmogorov is supported by the European Research Council under the European Unions Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement no 616160. Stanislav Živný is supported by a Royal Society University Research Fellowship.

[†]Institute of Science and Technology Austria, Austria (vnk@ist.ac.at)

[‡]Université Paris-Est, Marne-la-Vallée, France (thapper@u-pem.fr)

[§]University of Oxford, UK (standa.zivny@cs.ox.ac.uk)

of the instances [30, 58]. Another possibility, suggested by Feder and Vardi [25], is to restrict the constraint language; that is, all constraint relations in a given instance must belong to a fixed, finite set of relations on the domain. The most successful approach to classifying the language-restricted CSP is the so-called algebraic approach [6, 39, 40], which has led to several complexity classifications [1, 4, 5, 7] and algorithmic characterisations [3, 35] going beyond the seminal work of Schaefer [63].

1.2. Valued Constraint Satisfaction. The CSP deals with only feasibility issues: Is there a solution satisfying certain constraints? In this work we are interested in problems that capture both feasibility and optimisation issues: What is the best solution satisfying certain constraints? Problems of this form can be cast as valued constraint satisfaction problems [38, 76].

An instance of the *valued constraint satisfaction problem* (VCSP) is given by a collection of variables that must be assigned labels from a given domain with the goal to *minimise* the objective function that is given by the sum of cost functions, each depending on some subset of the variables [12]. The cost functions can take on finite rational values and positive infinity. The VCSP framework is very robust and has also been studied under different names such as Min-Sum problems, Gibbs energy minimisation, Markov Random Fields, Conditional Random Fields and others in different contexts in computer science [17, 56, 74].

The CSP corresponds to the special case of the VCSP when the codomain of all cost functions is $\{0, \infty\}$. Given a CSP instance, the Max-CSP consists in determining the maximum possible number of satisfied constraints, or equivalently with respect to exact solvability, the minimum number of unsatisfied constraints. The Max-CSP corresponds to the case of the VCSP when the codomain of all cost functions is $\{0, 1\}$.

The VCSP is NP-hard in general and thus we are interested in the restrictions which give rise to tractable classes of problems. As for the CSP, one can restrict the structure of the instances [28]. We will be interested in restricting the *valued constraint language*; that is, all cost functions in a given instance must belong to a fixed set of cost functions on the domain. The ultimate goal is to understand the computational complexity of all valued constraint languages, that is, determine which languages give rise to classes of problems solvable in polynomial time and which languages give rise to classes of problems that are NP-hard. Languages of the former type are called *tractable*, and languages of the latter type are called *intractable*.

Given the generality of the VCSP, it is not surprising that only few valued constraint languages have been completely classified as tractable or intractable. In particular, only Boolean (on a 2-element domain) languages [12, 18] and conservative (containing all $\{0, 1\}$ -valued unary cost functions) languages [50] have been completely classified with respect to exact solvability.

Extending the notion of (generalised) arc consistency for the CSP [26, 57] and several previously studied notions of arc consistencies for the VCSP [16], Cooper et al. introduced optimal soft arc consistency (OSAC) [14, 15], which is a linear programming relaxation of a given VCSP instance. In fact, the VCSP problem has a natural linear programming (LP) relaxation, proposed independently by a number of authors [8, 46, 52, 55, 67, 73, 75]. This relaxation is referred to as the *basic* LP relaxation (BLP) of VCSP as it is the first level in the Sherali-Adams hierarchy [66], which provides successively tighter LP relaxation of an integer LP. The BLP relaxation of a VCSP instance is known to be equivalent to the dual (Lagrangian) decomposition of the instance in which the subproblems are chosen as the individual constraints [41, 51, 68].

It is known that OSAC is at least as tight as BLP.¹

Apart from exact solvability of the CSP and its optimisation variants, the approximability of the Max-CSP has attracted a lot of attention [19, 42, 44]. Under the assumption of the unique games conjecture [45], Raghavendra has shown that the optimal approximation ratio for the finite-valued CSP is achieved by the basic semidefinite programming relaxation [61, 62]. Recently, the classes of the Max-CSP that are robustly approximable have been characterised [2, 21, 55]. Specifically, Kun et al. have studied the question of which classes of the Max-CSP can be robustly approximated using BLP [55]. Moreover, the power of BLP with respect to constant-factor approximation of finite-valued CSPs has been recently studied [20, 24].

More details on the complexity of the CSP can be found in [31] and more details on the complexity of the VCSP can be found in the recent survey [38].

1.3. Contributions. We study the power of the *basic linear programming relaxation* (BLP). Our main result is a precise characterisation of valued constraint languages for which BLP is a decision procedure. In other words, we characterise valued constraint languages over which VCSP instances can be solved *exactly* by the BLP, i.e., when the BLP has integrality gap 1.

The characterisation is algebraic in terms of *fractional polymorphisms* [10]. For a valued constraint language Γ with codomain the set of rationals with infinity, BLP is a decision procedure for Γ if and only if Γ admits a symmetric fractional polymorphism of *every* arity. For a valued constraint language Γ with codomain the set of rationals (so-called finite-valued languages), BLP is a decision procedure if and only if any of the following equivalent statements is satisfied: Γ admits a symmetric fractional polymorphism of *every* arity; Γ admits a symmetric fractional polymorphism of *some* arity; Γ admits a symmetric fractional polymorphism of *arity 2*; Γ admits a fractional polymorphism ω such that the support of ω generates a symmetric operation (possibly of different arity than the arity of ω).

Our work links solving VCSP instances exactly using linear programming and the algebraic machinery for the language-restricted VCSP introduced by Cohen et al. in [9, 13]. Part of the proof is inspired by the characterisation of the width-1 CSP [22, 25]. The two main technical contributions are the construction of a symmetric fractional polymorphism of a general-valued language (Theorem 2) and the construction of symmetric fractional polymorphisms of all arities of a finite-valued language (Theorem 4). In order to prove these two results, we present two techniques: a “tree cutting” argument, used in Section 6 to prove Theorem 2, and an argument based on a “graph of generalised operations”, used in Section 8 to prove Theorem 4.

Our results allow us to demonstrate that several valued constraint languages are tractable; that is, VCSP instances over these languages can be solved exactly using BLP. Languages not previously known to be tractable include: (1) submodular languages on *arbitrary lattices*; (2) k -submodular languages on *arbitrary finite domains* ($k = 2$ corresponds to bisubmodularity); (3) weakly (and hence strongly) tree-submodular languages on *arbitrary trees*. The complexity of (subclasses of) these languages has been mentioned explicitly as open problems in [23, 33, 48, 53].

1.4. Follow-up work. Since the announcement of our results [69], they have already been used to settle the complexity of the minimum 0-extension problem [32],

¹The difference between BLP and OSAC is that (the dual of) OSAC has only one variable for all constraints with the same scope (seen as a set) of variables. In BLP, different constraints yield different BLP variables even if the scopes (seen as sets) are the same.

the complexity of the 3-element finite-valued VCSP [34], and the complexity of the 3-element Min-Sol problems and conservative Min-Cost-Hom problems [71]. Moreover, the last two authors have recently shown that for finite-valued constraint languages, the condition of admitting a symmetric fractional polymorphism of arity 2 is also necessary for tractability [70].

1.5. Combinatorial Optimisation. Throughout the paper we assume that the objective function in our problem is represented as a sum of functions each defined on some subset of the variables. There is a rich tradition in combinatorial optimisation of studying problems in which the objective function to be optimised is represented by a value-giving *oracle*. In this model, a problem is tractable if it can be solved in polynomial time using only polynomially many queries to the oracle (where the polynomial is in the number of variables). Any query to the oracle can be easily simulated in linear time in the VCSP model. Consequently, a tractability result (for a class of functions) in the value oracle model automatically carries over to the VCSP model, while hardness results automatically carries over in the opposite direction.

One class of functions that has received particular attention in the value oracle model is the class of submodular functions. There are several known algorithms for minimising a (finite-valued) submodular function using only a polynomial number of calls to a value-giving oracle, see, for instance, Iwata’s survey [36]. Most previously discovered tractable valued constraint languages are somewhat related to submodular functions on distributive lattices [11, 12, 43, 50]. However, some VCSP instance with submodular functions can be solved much more efficiently than by using these general approaches [77].

Whilst submodular functions given by an oracle can be minimised in pseudopolynomial time on diamonds [54], in polynomial time on distributive lattices [37, 65] and the pentagon [53], and several constructions on lattices preserving tractability have been identified [53], it is an interesting open question as to what happens on non-distributive lattices. Similarly, k -submodular functions given by an oracle can be minimised in polynomial-time on domains of size three [27], but the complexity is unknown on domains of larger size [33]. It is known that strongly tree-submodular functions given by an oracle can be minimised in polynomial time on binary trees [48], but the complexity is open on general (non-binary) trees. Similarly, it is known that weakly tree-submodular functions given by an oracle can be minimised in polynomial time on chains and forks [48], but the complexity on (even binary) trees is open.

2. Background. In this section we describe the necessary background for the rest of the paper. We start with some basic notation. We denote $\mathbb{Q}_{\geq 0} = \{x \in \mathbb{Q} \mid x \geq 0\}$, $\overline{\mathbb{Q}} = \mathbb{Q} \cup \{\infty\}$ and $\overline{\mathbb{Q}}_{\geq 0} = \mathbb{Q}_{\geq 0} \cup \{\infty\}$. We define sets of real numbers $\mathbb{R}_{\geq 0}$, $\overline{\mathbb{R}}$ and $\overline{\mathbb{R}}_{\geq 0}$ in a similar way. Throughout the paper we assume that $0 \cdot \infty = 0$ and $x \cdot \infty = \infty$ for $x > 0$ (we will never use such multiplication in the case when $x < 0$). Note that value ∞ is understood as positive infinity (and accordingly $x < \infty$ for any $x \in \mathbb{R}$).

2.1. Valued CSP. Throughout the paper, let D be a fixed finite domain. We will call the elements of D *labels* (for variables). A function $f : D^m \rightarrow \overline{\mathbb{Q}}$ is called an m -ary cost function and we say that f takes *values*. The argument of f is called a *labelling*. For a cost function f , we denote $\text{dom } f = \{x \in D^m \mid f(x) < \infty\}$.

A *language* Γ is a set of cost functions of possibly different arities. A language Γ is called *finite-valued* if the codomain of every $f \in \Gamma$ is \mathbb{Q} . If Γ is not finite-valued we may emphasise this fact by calling Γ *general-valued*.

DEFINITION 1. An instance \mathcal{I} of the valued constraint satisfaction problem (VCSP)

is a function $D^V \rightarrow \overline{\mathbb{Q}}$ given by

$$f_{\mathcal{I}}(x) = \sum_{t \in T} f_t(x_{v(t,1)}, \dots, x_{v(t,n_t)}).$$

It is specified by a finite set of variables V , finite set of terms T , cost functions $f_t : D^{n_t} \rightarrow \overline{\mathbb{Q}}$ of arity n_t and indices $v(t, k) \in V$ for $t \in T, k = 1, \dots, n_t$. The indices of term $t \in T$ give the scope of the cost function f_t . A solution to \mathcal{I} is a labelling (also an assignment) $x \in D^V$ with the minimum total value. The instance \mathcal{I} is called a Γ -instance if all terms f_t belong to Γ .

The class of optimisation problems consisting of all Γ -instances is referred to as $\text{VCSP}(\Gamma)$. A language Γ is called tractable if $\text{VCSP}(\Gamma')$ can be solved in polynomial time for each finite $\Gamma' \subseteq \Gamma$. It is called NP-hard if $\text{VCSP}(\Gamma')$ is NP-hard for some finite $\Gamma' \subseteq \Gamma$.

2.2. The basic LP relaxation. Let \mathbb{M}_n be the set of probability distributions over labellings in D^n , i.e. $\mathbb{M}_n = \{\mu \geq 0 \mid \sum_{x \in D^n} \mu(x) = 1\}$. We also denote $\Delta = \mathbb{M}_1$; thus, Δ is the standard $(|D| - 1)$ -dimensional simplex. The corners of Δ can be identified with elements in D . For a distribution $\mu \in \mathbb{M}_n$ and a variable $v \in \{1, \dots, n\}$ let $\mu_{[v]} \in \Delta$ be the marginal probability of distribution μ for v :

$$\mu_{[v]}(a) = \sum_{x \in D^n: x_v = a} \mu(x) \quad \forall a \in D.$$

Given an instance \mathcal{I} , we define the value $\text{BLP}(\mathcal{I})$ as follows:

$$\begin{aligned} \text{BLP}(\mathcal{I}) &= \min \sum_{t \in T} \sum_{x \in \text{dom } f_t} \mu_t(x) f_t(x) & (2.1) \\ \text{s.t. } & (\mu_t)_{[k]} = \alpha_{v(t,k)} \quad \forall t \in T, k \in \{1, \dots, n_t\} \\ & \mu_t \in \mathbb{M}_{n_t} \quad \forall t \in T \\ & \mu_t(x) = 0 \quad \forall t \in T, x \notin \text{dom } f_t \\ & \alpha_v \in \Delta \quad \forall v \in V \end{aligned}$$

If there are no feasible solutions then $\text{BLP}(\mathcal{I}) = \infty$. All constraints in this system are linear, therefore this is a linear program. We call it the *basic LP relaxation* of \mathcal{I} (BLP). We say that BLP *solves* \mathcal{I} if $\text{BLP}(\mathcal{I}) = \min_{x \in D^n} f_{\mathcal{I}}(x)$. We say that BLP solves a language Γ if it solves all instances $\mathcal{I} \in \text{VCSP}(\Gamma)$.

2.3. Fractional polymorphisms. We denote by $\mathcal{O}^{(m)}$ the set of m -ary operations $g : D^m \rightarrow D$. An operation $g : D^2 \rightarrow D$ of arity 2 is called *binary*. An m -ary projection on the i th coordinate is the operation $e_i^{(m)} : D^m \rightarrow D$ defined by $e_i^{(m)}(x_1, \dots, x_m) = x_i$. Let S_m be the symmetric group on $\{1, \dots, m\}$. An operation $g \in \mathcal{O}^{(m)}$ is called *symmetric* if it is invariant with respect to any permutation of its arguments: $g(x_1, \dots, x_m) = g(x_{\pi(1)}, \dots, x_{\pi(m)})$ for any permutation $\pi \in S_m$ and any $(x_1, \dots, x_m) \in D^m$. The set of symmetric operations in $\mathcal{O}^{(m)}$ will be denoted by $\mathcal{O}_{\text{sym}}^{(m)}$.

A *fractional operation* of arity m is a vector $\omega : \mathcal{O}^{(m)} \rightarrow \mathbb{R}_{\geq 0}$ satisfying $\|\omega\|_1 = 1$, where $\|\omega\|_1 = \sum_{g \in \mathcal{O}^{(m)}} \omega(g)$. We let $\text{supp}(\omega)$ denote the support of ω , defined by $\text{supp}(\omega) = \{g \in \mathcal{O}^{(m)} \mid \omega(g) > 0\}$.

It will often be convenient to write a fractional operation $\omega : \mathcal{O}^{(m)} \rightarrow \mathbb{R}_{\geq 0}$ as a sum $\omega = \sum_{g \in \mathcal{O}^{(m)}} \omega(g) \cdot \chi_g$, where $\chi_g : \mathcal{O}^{(m)} \rightarrow \mathbb{R}_{\geq 0}$ denotes the vector that assigns weight 1 to the operation g and 0 to all other operations.

A fractional operation ω is called symmetric if all operations in $\text{supp}(\omega)$ are symmetric.

The *superposition*, $h[g_1, \dots, g_n]$, of an n -ary operation h with n m -ary operations g_1, \dots, g_n is the m -ary operation defined by

$$h[g_1, \dots, g_n](x_1, \dots, x_m) = h(g_1(x_1, \dots, x_m), \dots, g_n(x_1, \dots, x_m)).$$

This can also be seen as a composition $h \circ (g_1, \dots, g_n) : D^m \rightarrow D$ of the operation $h : D^n \rightarrow D$ and the mapping $(g_1, \dots, g_n) : D^m \rightarrow D^n$.

The *superposition*, $\omega[g_1, \dots, g_n]$, of an n -ary fractional operation ω with n m -ary operations g_1, \dots, g_n is the m -ary fractional operation defined as follows:

$$\omega[g_1, \dots, g_n](h) = \sum_{\{h' \mid h=h'[g_1, \dots, g_n]\}} \omega(h').$$

The following example illustrates these definitions.

EXAMPLE 1. Let $D = \{0, 1, \dots, d\}$ and let $\min, \max : D^2 \rightarrow D$ be the two binary operations on D that return the smaller (larger) of its two arguments respectively with respect to the natural order of integers.

The ternary operation $\min^{(3)}$ returning the smallest of its three arguments can be obtained by the following superposition: $\min^{(3)}(x, y, z) = \min[e_1^{(3)}, \min[e_2^{(3)}, e_3^{(3)}]]$.

Let ω be the fractional operation that assigns weight $\frac{1}{2}$ to \min and weight $\frac{1}{2}$ to \max . Clearly, $\text{supp}(\omega) = \{\min, \max\}$. Since both \min and \max are symmetric operations, we have that ω is a symmetric fractional operation.

Let $\min_{12}^{(4)}$ and $\min_{34}^{(4)}$ be the two 4-ary operations that return the smaller of its first (last) two arguments respectively. Then the superposition of ω with $\min_{12}^{(4)}$ and $\min_{34}^{(4)}$ is the 4-ary fractional operation $\omega' = \omega[\min_{12}^{(4)}, \min_{34}^{(4)}]$ that assigns weight $\frac{1}{2}$ to the operation $\min^{(4)}$, which returns the smallest of its four arguments, and weight $\frac{1}{2}$ to the operation $\max[\min_{12}^{(4)}, \min_{34}^{(4)}]$.

DEFINITION 2. For an n -ary cost function $f : D^n \rightarrow \overline{\mathbb{Q}}$ and $x^1, \dots, x^m \in D^n$ for some $m \geq 1$, we define the average value of f applied to the labellings x^1, \dots, x^m by

$$f^m(x^1, \dots, x^m) = \frac{1}{m}(f(x^1) + \dots + f(x^m)).$$

DEFINITION 3. A fractional operation $\omega : \mathcal{O}^{(m)} \rightarrow \mathbb{R}_{\geq 0}$ is called a fractional polymorphism of the language Γ , and we say that Γ admits ω , if for every cost function $f \in \Gamma$,

$$\sum_{g \in \mathcal{O}^{(m)}} \omega(g)f(g(x^1, \dots, x^m)) \leq f^m(x^1, \dots, x^m) \quad \forall x^1, \dots, x^m \in \text{dom } f. \quad (2.2)$$

We note that (2.2) implies that if $g \in \text{supp}(\omega)$ and $x^1, \dots, x^m \in \text{dom } f$ then $g(x^1, \dots, x^m) \in \text{dom } f$.

Definition 3 is illustrated in Figure 2.1, which should be read from left to right. Let f be an n -ary cost function and $\omega : \mathcal{O}^{(m)} \rightarrow \mathbb{R}_{\geq 0}$ an m -ary fractional operation. Moreover, let $k = |\mathcal{O}^{(m)}|$. Starting with m n -tuples $x^1, \dots, x^m \in \text{dom } f$, which we view as row vectors in Figure 2.1, we first apply all m -ary operations g_1, \dots, g_k to these tuples componentwise, thus obtaining the m -tuples y^1, \dots, y^k . Inequality 2.2 amounts

$$\begin{array}{ccc}
\begin{array}{l} x^1 \\ x^2 \\ \vdots \\ x^m \end{array} & = \begin{array}{l} (x_1^1 \ x_2^1 \ \dots \ x_n^1) \\ (x_1^2 \ x_2^2 \ \dots \ x_n^2) \\ \vdots \\ (x_1^m \ x_2^m \ \dots \ x_n^m) \end{array} & \xrightarrow{f} \left. \begin{array}{l} f(x^1) \\ f(x^2) \\ \vdots \\ f(x^m) \end{array} \right\} \frac{1}{m} \sum_{i=1}^m f(x^i) \\
\begin{array}{l} y^1 = g_1(x^1, \dots, x^m) \\ y^2 = g_2(x^1, \dots, x^m) \\ \vdots \\ y^k = g_k(x^1, \dots, x^m) \end{array} & = \begin{array}{l} (y_1^1 \ y_2^1 \ \dots \ y_n^1) \\ (y_1^2 \ y_2^2 \ \dots \ y_n^2) \\ \vdots \\ (y_1^k \ y_2^k \ \dots \ y_n^k) \end{array} & \xrightarrow{f} \left. \begin{array}{l} f(y^1) \\ f(y^2) \\ \vdots \\ f(y^k) \end{array} \right\} \sum_{i=1}^k \omega(g_i) f(y^i)
\end{array}$$

FIGURE 2.1. Definition of a fractional polymorphism.

to comparing the average of the values of f applied to the tuples x^1, \dots, x^m with the weighted sum of the values of f applied to the tuples y^1, \dots, y^k , where the weight of the i th tuple y^i (obtained from g_i) is the weight assigned to g_i by ω .

EXAMPLE 2. A simple example of an m -ary fractional operation is the vector $\omega : \mathcal{O}^{(m)} \rightarrow \mathbb{R}_{\geq 0}$ defined by $\omega(e_i^{(m)}) = 1/m$ for all $1 \leq i \leq m$ and $\omega(h) = 0$ for any m -ary operation h that is not a projection. It follows from Definition 3 that ω is a fractional polymorphism of every cost function f and in fact (2.2) holds with equality in this case.

EXAMPLE 3. Recall from Example 1 the fractional operation ω defined on D that assigns weight $\frac{1}{2}$ to \min and weight $\frac{1}{2}$ to \max . In this special case, (2.2) simplifies to

$$f(\min(x^1, x^2)) + f(\max(x^1, x^2)) \leq f(x^1) + f(x^2) \quad \forall x^1, x^2 \in \text{dom } f.$$

For $D = \{0, 1\}$, a cost function f that satisfies (2.2) with this ω is called submodular [65].

REMARK 1. One can equivalently view fractional polymorphisms in a probabilistic setting. A fractional operation $\omega : \mathcal{O}^{(m)} \rightarrow \mathbb{R}_{\geq 0}$ is a fractional polymorphism of Γ if ω is a probability distribution over $\mathcal{O}^{(m)}$, and every cost function $f \in \Gamma$ satisfies,

$$\mathbb{E}_{g \sim \omega} f(g(x^1, \dots, x^m)) \leq f^m(x^1, \dots, x^m) \quad \forall x^1, \dots, x^m \in \text{dom } f.$$

3. Results. In this section we will state our main results together with some algorithmic consequences. The rest of the paper will be devoted to the proofs of the results.

3.1. The power of BLP. Let Γ be a language such that the set Γ is countable. First, we give a precise characterisation of the power of BLP for (general-valued) languages.

THEOREM 1. BLP solves Γ if and only if Γ admits a symmetric fractional polymorphism of every arity $m \geq 2$.

Second, we give a sufficient condition for the existence of symmetric fractional polymorphisms. But first we need a standard definition from universal algebra.

A set \mathcal{C} of operations is called a *clone* if it contains all projections and is closed under superposition; that is, \mathcal{C} contains $e_i^{(m)}$ for all $m \geq 1$ and $1 \leq i \leq m$, and if $h, g_1, \dots, g_n \in \mathcal{C}$ then $h[g_1, \dots, g_n] \in \mathcal{C}$, where h is an n -ary operation and g_1, \dots, g_n

are operations of the same arity. A set \mathcal{O} of operations is said to *generate* g if g belongs to the smallest clone containing \mathcal{O} ; in other words, g can be obtained by superpositions of operations from \mathcal{O} and projections.

EXAMPLE 4. *If \mathcal{O} contains the binary maximum operation $\max : D^2 \rightarrow D$ that returns the larger of its two arguments (with respect to some total order on D), then \mathcal{O} can generate the m -ary operation $\max^{(m)}$ that returns the largest of its m arguments by $\max^{(m)}(x_1, \dots, x_m) = \max(x_1, \max(x_2, \dots, \max(x_{m-1}, x_m) \dots))$.*

THEOREM 2. *Suppose that, for every $n \geq 2$, Γ admits a fractional polymorphism ω_n such that $\text{supp}(\omega_n)$ generates a symmetric n -ary operation. Then, Γ admits a symmetric fractional polymorphism of every arity $m \geq 2$.*

The following is an immediate consequence of Theorems 1 and 2.

COROLLARY 3. *BLP solves Γ if and only if for every $n \geq 2$, Γ admits a fractional polymorphism ω_n such that $\text{supp}(\omega_n)$ generates a symmetric n -ary operation.*

Finally, we give, in Theorem 5, a more refined characterisation of the power of BLP for finite-valued languages. It is based on the following result.

THEOREM 4. *Suppose that a finite-valued language Γ admits a symmetric fractional polymorphism of arity $m - 1 \geq 2$. Then Γ admits a symmetric fractional polymorphism of arity m .*

THEOREM 5. *Suppose that Γ is finite-valued. The following are equivalent:*

1. *BLP solves Γ ;*
2. *Γ admits a symmetric fractional polymorphism of every arity $m \geq 2$;*
3. *Γ admits a symmetric fractional polymorphism of some arity $m \geq 2$;*
4. *Γ admits a symmetric fractional polymorphism of arity 2;*
5. *For every $n \geq 2$, Γ admits a fractional polymorphism ω_n such that $\text{supp}(\omega_n)$ generates a symmetric n -ary operation.*

Proof. The equivalence between statements (1) and (2) is a special case of Theorem 1. The implications (2) \implies (4) \implies (3) and (2) \implies (5) are trivial. The implication (4) \implies (2) follows from Theorem 4. Assume (3). By Theorem 4, we may assume that m is even. Let $e_1^{(2)}$ and $e_2^{(2)}$ be the two binary projections on the domain of Γ . Then $\omega[e_1^{(2)}, e_2^{(2)}, \dots, e_1^{(2)}, e_2^{(2)}]$ is a binary symmetric fractional polymorphism of Γ , so (4) follows. Finally, the implication (5) \implies (3) follows from Theorem 2. \square

Note that the finite-valuedness assumption in Theorems 4 and 5 is essential: for general-valued languages these theorems do not hold as the following example demonstrates.

EXAMPLE 5. *Let $D = \{a, b, c\}$ and consider the binary operation $g : D^2 \rightarrow D$ defined by $g(x, x) = x$ for $x \in D$, $g(a, b) = g(b, a) = b$, $g(b, c) = g(c, b) = c$, and $g(a, c) = g(c, a) = a$ (g corresponds to the oriented cycle $a \rightarrow b \rightarrow c \rightarrow a$). Note that g is symmetric. Moreover, g is also conservative, that is, $g(x, y) \in \{x, y\}$ for all $x, y \in D$. Any operation that is symmetric and conservative is called a tournament operation [11]. Consider the fractional operation ω defined by $\omega(g) = 1$. It is known that any general-valued constraint language admitting ω is tractable [11] (ω is called a tournament pair in [11]).*

Let $f : D^2 \rightarrow \mathbb{Q}_{\geq 0}$ be the following binary cost function: $f(x, y) = 0$ if $(x, y) \in \{(a, b), (b, c), (c, a)\}$ and $f(x, y) = \infty$ otherwise. Let $\Gamma = \{f\}$. It can be verified that Γ admits ω as a fractional polymorphism and thus is tractable.

We now show, however, that Γ does not admit any ternary symmetric fractional polymorphism. Let $h : D^3 \rightarrow D$ be an arbitrary ternary symmetric operation. Since $\text{dom } f = \{(a, b), (b, c), (c, a)\}$, we have that h applied to the tuples (a, b) , (b, c) , and (c, a) componentwise gives a tuple (x, x) for some $x \in D$ but $(x, x) \notin \text{dom } f$ for any

$x \in D$. Thus no ternary fractional polymorphism of Γ can have a symmetric ternary operation in its support. By Theorem 1, BLP does not solve Γ .

3.2. Examples of languages solved by BLP. We now give examples of languages that are solved by BLP. In some cases, the tractability of these languages was known before, while in others, we present here the first proof of their tractability.

A binary operation $g : D^2 \rightarrow D$ is *idempotent* if $g(x, x) = x$ for all $x \in D$, *commutative* if $g(x, y) = g(y, x)$ for all $x, y \in D$, and *associative* if $g(x, g(y, z)) = g(g(x, y), z)$ for all $x, y, z \in D$. A binary operation $g : D^2 \rightarrow D$ is a *semilattice operation* if g is idempotent, commutative, and associative. The max-operation of Example 4 is an example of a semilattice operation. In the same way that max generates $\max^{(m)}$, $m \geq 2$, every semilattice operation $g : D^2 \rightarrow D$ generates symmetric operations of all arities. In particular, a symmetric operation $g^{(m)} : D^m \rightarrow D$ can be obtained from g by

$$g^{(m)}(x_1, \dots, x_m) = g(x_1, g(x_2, \dots, g(x_{m-1}, x_m) \dots)).$$

Consequently, we obtain the following result.

COROLLARY 6 (of Theorem 1 and Theorem 2). *If Γ admits a fractional polymorphism with a semilattice operation in its support, then BLP solves Γ .*

Most previously identified tractable languages have been defined via *binary multimorphisms*, which are a special case of binary fractional polymorphisms [12]. A binary multimorphism $\langle g_1, g_2 \rangle$ of a language Γ is a binary fractional polymorphism ω of Γ such that $\omega(g_1) = \omega(g_2) = 1/2$, where $g_1, g_2 : D^2 \rightarrow D$. For a binary multimorphism $\langle g_1, g_2 \rangle$, the fractional polymorphism inequality (2.2) simplifies to,

$$f(g_1(x^1, x^2)) + f(g_2(x^1, x^2)) \leq f(x^1) + f(x^2) \quad \forall f \in \Gamma, x^1, x^2 \in \text{dom } f.$$

With the exception of skew bisubmodularity, the languages discussed below are all defined by binary multimorphisms.

Submodularity on a lattice. Let $(D; \wedge, \vee)$ be an *arbitrary* lattice on D , where \wedge and \vee are the meet and join operations, respectively. Let Γ be a language admitting the multimorphism $\langle \wedge, \vee \rangle$; such languages are called *submodular* on the lattice $(D; \wedge, \vee)$. The operations \wedge and \vee of any lattice are semilattice operations, hence Corollary 6 shows that BLP solves Γ . The tractability of submodular languages was previously known only for distributive lattices [37, 65]. Moreover, several tractability-preserving operations on lattices have been identified in [53]. Finally, it is known that VCSP instances over submodular languages on diamonds can be minimised in pseudopolynomial time [54].

Symmetric tournament pair. A binary operation $g : D^2 \rightarrow D$ is *conservative* if $g(x, y) \in \{x, y\}$ for all $x, y \in D$. A binary operation $g : D^2 \rightarrow D$ is a *tournament operation* if g is commutative and conservative. The dual of a tournament operation g is the unique tournament operation g' satisfying $g(x, y) \neq g'(x, y)$ for all $x \neq y$. The multimorphism $\langle g_1, g_2 \rangle$ is a *symmetric tournament pair* (STP) if both g_1 and g_2 are tournament operations and g_2 is the dual of g_1 [11]. If Γ is a finite-valued language with an STP multimorphism $\langle g_1, g_2 \rangle$ then Γ also admits a submodularity multimorphism discussed above. This result is implicitly contained in [11] and a full proof is given in Appendix A. Consequently, BLP solves Γ by Corollary 6. This also follows from Theorem 5.

k -Submodularity. Let $D = \{0, 1, \dots, k\}$ and let Γ be a language defined on D that admits the multimorphism $\langle \min_0, \max_0 \rangle$ [12], where $\min_0(x, x) = x$ for all $x \in D$ and $\min_0(x, y) = 0$ for all $x, y \in D, x \neq y$; $\max_0(x, y) = 0$ if $0 \neq x \neq y \neq 0$ and $\max_0(x, y) = \max(x, y)$ otherwise, where \max returns the larger of its two arguments with respect to the normal order of integers; such languages are known as *k-submodular* [33]. Since \min_0 is a semilattice operation, BLP solves Γ by Corollary 6. The tractability of k -submodular languages was previously open for $k > 2$ [33].

Applications of k -submodular functions can be found in [29, 72].

Bisubmodularity. The special case of k -submodularity for $k = 2$ is known as *bisubmodularity*. The tractability of (finite-valued) bisubmodular languages was previously known only using a general algorithm for minimising bisubmodular set functions [27, 59].

Skew bisubmodularity. Let $D = \{0, 1, 2\}$ with the partial order satisfying $0 < 1$ and $0 < 2$. Recall the definition of the operations \min_0 and \max_0 from the description of k -submodularity above. We define $\max_1(x, y) = 1$ if $0 \neq x \neq y \neq 0$ and $\max_1(x, y) = \max(x, y)$ otherwise, where \max returns the larger of its two arguments with respect to the normal order of integers. A language Γ defined on D is called *α -bisubmodular*, for some real $0 < \alpha \leq 1$, if Γ admits a fractional polymorphism ω defined by $\omega(\min_0) = 1/2$, $\omega(\max_0) = \alpha/2$, and $\omega(\max_1) = (1 - \alpha)/2$. (Note that 1-bisubmodular languages are bisubmodular languages discussed above.) A language that is α -bisubmodular for some α is called *skew bisubmodular*. Since \min_0 is a semilattice operation, BLP solves Γ by Corollary 6. The tractability of skew bisubmodular languages was first observed in [34] using an extended abstract of this paper [69].

Strong tree-submodularity. Assume that the labels in the domain D are arranged into a tree T . The tree induces a partial order: $a \preceq b$ if a is an ancestor of b , that is, if a lies on the unique path from b to the root of T . Given $a, b \in T$, let P_{ab} denote the unique path in T between a and b of length (=number of edges) $d(a, b)$, and let $P_{ab}[i]$ denote the i -th vertex on P_{ab} , where $0 \leq i \leq d(a, b)$ and $P_{ab}[0] = a$. Let $\langle g_1, g_2 \rangle$ be two binary commutative operations defined as follows: given a and b , let $a_1 = P_{ab}[\lfloor d/2 \rfloor]$ and $a_2 = P_{ab}[\lceil d/2 \rceil]$. If $a_2 \preceq a_1$ then swap a_1 and a_2 so that $a_1 \preceq a_2$. Finally, $g_1(a, b) = g_1(b, a) = a_1$ and $g_2(a, b) = g_2(b, a) = a_2$. Let Γ be a language admitting the multimorphism $\langle g_1, g_2 \rangle$; such languages are called *strongly tree-submodular*. Since g_1 is a semilattice operation, BLP solves Γ by Corollary 6. The tractability of finite-valued strongly tree-submodular languages on binary trees has been shown in [48] but the tractability of strongly tree-submodular languages on non-binary trees was left open.

Weak tree-submodularity. Assume that the labels in the domain D are arranged into a tree T . For $a, b \in T$, let $g_1(a, b)$ be defined as the highest common ancestor of a and b in T ; that is, the unique node on the path P_{ab} that is ancestor of both a and b . We define $g_2(a, b)$ as the unique node on the path P_{ab} such that the distance between a and $g_2(a, b)$ is the same as the distance between b and $g_1(a, b)$. Let Γ be a language admitting the multimorphism $\langle g_1, g_2 \rangle$; such languages are called *weakly tree-submodular*. Since g_1 is a semilattice operation, BLP solves Γ by Corollary 6. The

tractability of finite-valued weakly tree-submodular languages on chains² and forks³ has been shown in [48] and left open for all other trees. Weak tree-submodularity generalises the above-discussed concept of strong tree-submodularity in the sense that any language that is strongly tree-submodular is also weakly tree-submodular [48]. Weak tree-submodularity also generalises the above-discussed concept of k -submodularity, which corresponds to the special case with a tree on $k + 1$ vertices consisting of a root node with k children.

1-Defect chain. In our final example, Corollary 6 does not suffice to prove that BLP solves the specific languages. Instead, we refer directly to Theorems 1 and 2 (and also Theorem 5 in the special case of finite-valued languages). Let b and c be two distinct elements of D and let $(D; <)$ be a partial order which relates all pairs of elements except for b and c . A pair $\langle g_1, g_2 \rangle$, where $g_1, g_2 : D^2 \rightarrow D$ are two binary operations, is a *1-defect chain* multimorphism if g_1 and g_2 are both commutative and satisfy the following conditions:

- If $\{x, y\} \neq \{b, c\}$, then $g_1(x, y) = x \wedge y$ and $g_2(x, y) = x \vee y$.
- If $\{x, y\} = \{b, c\}$, then $\{g_1(x, y), g_2(x, y)\} \cap \{x, y\} = \emptyset$, and $g_1(x, y) < g_2(x, y)$.

The tractability of finite-valued languages admitting a 1-defect chain multimorphism has been shown in [43]. By Theorem 5, the BLP solves any such language. We now show a more general result: BLP also solves general-valued languages admitting a 1-defect chain multimorphism.

We consider the case when $g_1(b, c) < b, c$ and set $g = g_1$. (An analogous argument works in the case when $g_2(b, c) > b, c$.) Using g , we construct a symmetric m -ary operation $h^{(m)}(x_1, \dots, x_m)$ for each m . Consequently, BLP solves Γ by Theorem 1 and Theorem 2.

Let h_1, \dots, h_M be the $M = \binom{m}{2}$ terms $g(x_i, x_j)$. Let

$$h^{(m)} = g(h_1, g(h_2, \dots, g(h_{M-1}, h_M) \dots)).$$

There are three possible cases:

- $\{b, c\} \not\subseteq \{x_1, \dots, x_m\}$. Then g acts as \wedge , which is a semilattice operation, hence so does $h^{(m)}$.
- $\{b, c\} \subseteq \{x_1, \dots, x_m\}$ and $g(b, c) \leq x_1, \dots, x_m$. Then $h_i = g(b, c)$ for some $1 \leq i \leq M$, and $g(h_i, h_j) = g(b, c)$ for all $1 \leq j \leq M$, so $h^{(m)}(x_1, \dots, x_m) = g(b, c)$.
- $\{b, c\} \subseteq \{x_1, \dots, x_m\}$ and $x_p \leq g(b, c)$, for some $1 \leq p \leq m$. By choice of g , $x_p \notin \{b, c\}$ and we can additionally choose p so that $x_p \leq x_1, \dots, x_m$. Then $g(x_p, x_q) = x_p$ for all $1 \leq q \leq m$ so $h_i = x_p$ for some $1 \leq i \leq M$ and $g(h_i, h_j) = x_p$ for all $1 \leq j \leq M$, so $h^{(m)}(x_1, \dots, x_m) = x_p$.

3.3. Finding a solution. Let \mathcal{I} be an instance of $\text{VCSP}(\Gamma)$ and assume that the BLP solves Γ . We will now justify this terminology by showing how to obtain an actual assignment that optimises \mathcal{I} .

The basic idea is that of self-reduction: we iteratively assign labels to the first variable of \mathcal{I} and test whether the partially assigned instance has the same optimum as \mathcal{I} . When such a label is found, we proceed with the next variable. After $n \cdot |D|$ steps, where n is the number of variables of \mathcal{I} , we are guaranteed to have found an optimal assignment. This method requires that we can find the optimum of a partially

²A chain is a binary tree in which all nodes except leaves have exactly one child.

³A fork is a binary tree in which all nodes except leaves and one special node have exactly one child. The special node has exactly two children.

assigned instance. In order to do this, we need the following technical lemma which is proved in Section 7.

LEMMA 7. *There exists a subset $D' \subseteq D$ such that if Γ admits an m -ary symmetric fractional polymorphism, then it admits an m -ary symmetric fractional polymorphism ω such that, for all $g \in \text{supp}(\omega)$,*

1. $g(x, x, \dots, x) \in D'$ for all $x \in D$;
2. $g(x, x, \dots, x) = x$ for all $x \in D'$.

The utility of Lemma 7 can be described as follows: whenever we have an appropriate fractional polymorphism that maps into a sub-domain D' and that is idempotent on D' , then we can infer that there is an optimal solution just consisting of labels from D' . Such languages are made no more complex by adding constants restricting variables to be particular labels from D' . In particular, we can use BLP to solve partially assigned instances.

PROPOSITION 8. *Let Γ be an arbitrary valued constraint language and let \mathcal{I} be an instance of $\text{VCSP}(\Gamma)$. If BLP solves Γ , then an optimal assignment of \mathcal{I} can be found in polynomial time.*

Proof. Let $D' \subseteq D$ be the set in Lemma 7. Let c_d be the unary cost function with $c_d(x) = \infty$ for $x \neq d$ and $c_d(d) = 0$, and let $\Gamma_c = \Gamma \cup \{c_d \mid d \in D'\}$. It follows from Theorem 1 and Lemma 7(2) that the BLP solves Γ_c . We can now apply self-reduction to obtain an optimal assignment of \mathcal{I} . Let $x = (x_1, \dots, x_n)$ be the variables of \mathcal{I} . The idea is to successively try each possible label $d \in D'$ for x_1 by adding the term $c_d(x_1)$ to the sum of \mathcal{I} . The modified instance is an instance of $\text{VCSP}(\Gamma_c)$ so we can use the BLP to obtain its optimum. If, for some $d_1 \in D'$, the optimum of the modified instance matches that of \mathcal{I} , then we know that there exists an optimal solution to \mathcal{I} in which x_1 is assigned d_1 and we can proceed with the next variable.

We now claim that this procedure always terminates with an optimal assignment. In particular, we must show that if the optimum of \mathcal{I} is equal to the optimum of the instance \mathcal{I}' obtained by adding $\sum_{i=1}^k c_{d_i}(x_i)$, $d_i \in D'$, $k \geq 1$, to the sum of \mathcal{I} , then we can always find an optimal solution of \mathcal{I}' that assigns d_{k+1} to x_{k+1} , for some $d_{k+1} \in D'$. Let $s : V \rightarrow D$ be an optimal solution to \mathcal{I}' . Note that $s(x_i) = d_i$ for all $i = 1, \dots, k$. Let ω be a fractional polymorphism of Γ satisfying (1) and (2) in Lemma 7. Then, with \mathcal{I} represented as in Definition 1 on page 4, and with the notation $x^t = (x_{v(t,1)}, \dots, x_{v(t,n_t)})$,

$$\begin{aligned} f_{\mathcal{I}'}(s(x)) &= f_{\mathcal{I}}(s(x)) + \sum_{i=1}^k c_{d_i}(s(x_i)) \\ &\geq \sum_{t \in T} \sum_{g \in \text{supp}(\omega)} \omega(g) f_t(g(s(x^t), \dots, s(x^t))) \\ &= \sum_{g \in \text{supp}(\omega)} \omega(g) f_{\mathcal{I}}(g[s, \dots, s](x)) \geq \min_{x \in D^n} f_{\mathcal{I}}(x). \end{aligned}$$

Since $f_{\mathcal{I}'}(s(x)) = \min_{x \in D^n} f_{\mathcal{I}}(x)$, we conclude that $g[s, \dots, s]$ is an optimal solution to \mathcal{I} , for each $g \in \text{supp}(\omega)$. Every $g \in \text{supp}(\omega)$ satisfies (2) in Lemma 7, so $g[s, \dots, s](x_i) = d_i$ for all $i = 1, \dots, k$, and it follows that $g[s, \dots, s]$ is also an optimal solution to \mathcal{I}' . Finally, g satisfies (1) in Lemma 7, so $g[s, \dots, s](x_{k+1}) \in D'$, from which the claim follows. \square

4. Characterisation of general-valued languages. In this section we will prove the main characterisation of general-valued languages solved by BLP.

THEOREM 1 (restated). *BLP solves Γ if and only if Γ admits a symmetric fractional polymorphism of every arity $m \geq 2$.*

Recall that \mathbb{M}_n is the set of probability distributions over labellings in D^n and $\Delta = \mathbb{M}_1$. For an integer $m \geq 1$ we denote by $\mathbb{M}_n^{(m)}$ the set of vectors $\mu \in \mathbb{M}_n$ such that all components of μ are rational numbers of the form p/m , where $p \in \mathbb{Z}$. We also denote $\Delta^{(m)} = \mathbb{M}_1^{(m)}$, and define $\Omega^{(m)}$ to be the set of mappings $\Delta^{(m)} \rightarrow D$.

A vector $\alpha \in \Delta^{(m)}$ can be viewed as a multiset over D of size m , or equivalently as an element of the quotient space D^m / \sim where \sim is the equivalence relation defined by permutations. Therefore, a symmetric mapping $g : D^m \rightarrow D$ can be equivalently viewed as a mapping $g : \Delta^{(m)} \rightarrow D$. This gives a natural isomorphism between $\mathcal{O}_{\text{sym}}^{(m)}$ and $\Omega^{(m)}$.

A symmetric fractional polymorphism ω can thus be viewed either as a probability distribution over $\mathcal{O}_{\text{sym}}^{(m)}$ or as a probability distribution over $\Omega^{(m)}$. In the proposition below we use both views interchangeably.

PROPOSITION 9. *Let ω be a symmetric fractional operation of arity m and let f be a cost function of arity n . Then ω is a fractional polymorphism of f (i.e., (2.2) holds) if and only if for every $\mu \in \mathbb{M}_n^{(m)}$,*

$$\mathbb{E}_{g \sim \omega} f(g(\mu_{[1]}, \dots, \mu_{[n]})) \leq \mathbb{E}_{x \sim \mu} f(x), \quad (4.1)$$

where $\mu_{[i]} \in \Delta^{(m)}$ is the marginal probability of distribution μ for the i th coordinate.

Proof. The left-hand side of (4.1) is equal $\sum_{g \in \Omega^{(m)}} \omega(g) f(g(\mu_{[1]}, \dots, \mu_{[n]}))$ and the right-hand side is equal $\sum_{x \in D^n} \mu(x) f(x)$. The claim thus follows from Definition 3 and Remark 1. \square

Proof. (of Theorem 1) To prove Theorem 1, we need to establish the following: BLP solves Γ if and only if for every $m \geq 2$ there exists a probability distribution ω over $\Omega^{(m)}$ that satisfies (4.1).

“ \Leftarrow ”: Suppose that Γ admits a symmetric fractional polymorphism of every arity. Let \mathcal{I} be a Γ -instance with n variables. We need to show that system (2.1) for \mathcal{I} has an integral minimiser.

It is well known that every LP with rational coefficients has an optimal solution with rational coefficients (for instance, this is a direct consequence of Fourier-Motzkin elimination) [64]. Therefore, since the LP (2.1) has rational coefficients, it has an optimal solution $\{\alpha, \mu_t\}$ such that all variables are rational numbers of the form p/m for some integers p, m with $m \geq 1$. We can assume that $m \geq 2$, otherwise the claim is trivial.

Let ω be a symmetric fractional polymorphism of Γ of arity m . For any n -ary cost function f and $\mu \in \mathbb{M}_n^{(m)}$, we denote by $f(\mu)$ the expectation $f(\mu) = \mathbb{E}_{x \sim \mu} f(x)$.

Using (2.1) and (4.1), we can write

$$\begin{aligned} \text{BLP}(\mathcal{I}) &= \sum_{t \in T} \sum_{x \in \text{dom } f_t} \mu_t(x) f_t(x) = \sum_{t \in T} f_t(\mu_t) \\ &\geq \sum_{t \in T} \sum_{g \in \Omega^{(m)}} \omega(g) f(g(\mu_{v(t,1)}, \dots, \mu_{v(t,n_t)})) \\ &= \sum_{g \in \Omega^{(m)}} \omega(g) f_{\mathcal{I}}(g(\mu)), \end{aligned} \quad (4.2)$$

where g is applied component-wise, i.e., if $\alpha = (\alpha_1, \dots, \alpha_n) \in [\Delta^{(m)}]^n$, then $g(\alpha) = (g(\alpha_1), \dots, g(\alpha_n)) \in D^n$. (Note that in the second equality in (4.2) we have used

$0 \cdot \infty = 0$.) Eq. (4.2) implies that $\text{BLP}(\mathcal{I}) \geq f_{\mathcal{I}}(g(\mu))$ for some $g \in \text{supp}(\omega)$, and therefore BLP solves the instance \mathcal{I} .

“ \Rightarrow ”: Let us fix $m \geq 2$, and assume that BLP solves Γ . In this part we will use letters with a “hat” ($\hat{\alpha}$ and $\hat{\mu}$) for vectors of the form p/m , $p \in \mathbb{Z}$.

First, we consider the case when $|\Gamma|$ is finite. Suppose that Γ does not admit a symmetric fractional polymorphism of arity m . Using the notation

- Γ^+ is the set of tuples $(f, \hat{\mu}, \hat{\alpha})$ such that f is a function in Γ of arity n , $\hat{\mu} \in \mathbb{M}_n^{(m)}$ with $\text{supp}(\hat{\mu}) \subseteq \text{dom } f$, and $\hat{\alpha} = (\hat{\mu}_{[1]}, \dots, \hat{\mu}_{[n]}) \in [\Delta^{(m)}]^n$; and
- $\Omega_{\Gamma}^{(m)} \subseteq \Omega^{(m)}$ is the set of mappings $g : \Delta^{(m)} \rightarrow D$ such that $g(\hat{\alpha}) \in \text{dom } f$ for all $(f, \hat{\mu}, \hat{\alpha}) \in \Gamma^+$,

the following system does not have a solution:

$$\sum_{g \in \Omega_{\Gamma}^{(m)}} \omega(g) f(g(\hat{\alpha})) \leq f(\hat{\mu}) \quad \forall (f, \hat{\mu}, \hat{\alpha}) \in \Gamma^+ \quad (4.3a)$$

$$\sum_{g \in \Omega_{\Gamma}^{(m)}} \omega(g) = 1 \quad (4.3b)$$

$$\omega(g) \geq 0 \quad \forall g \in \Omega_{\Gamma}^{(m)} \quad (4.3c)$$

Since system (4.3) is infeasible, by Farkas’ lemma [64] the following system has a solution:

$$\sum_{(f, \hat{\mu}, \hat{\alpha}) \in \Gamma^+} y(f, \hat{\mu}, \hat{\alpha}) f(\hat{\mu}) + z < 0 \quad (4.4a)$$

$$\sum_{(f, \hat{\mu}, \hat{\alpha}) \in \Gamma^+} y(f, \hat{\mu}, \hat{\alpha}) f(g(\hat{\alpha})) + z \geq 0 \quad \forall g \in \Omega_{\Gamma}^{(m)} \quad (4.4b)$$

$$y(f, \hat{\mu}, \hat{\alpha}) \geq 0 \quad \forall (f, \hat{\mu}, \hat{\alpha}) \in \Gamma^+ \quad (4.4c)$$

Eliminating z gives

$$\sum_{(f, \hat{\mu}, \hat{\alpha}) \in \Gamma^+} y(f, \hat{\mu}, \hat{\alpha}) f(g(\hat{\alpha})) > \sum_{(f, \hat{\mu}, \hat{\alpha}) \in \Gamma^+} y(f, \hat{\mu}, \hat{\alpha}) f(\hat{\mu}) \quad \forall g \in \Omega_{\Gamma}^{(m)} \quad (4.5a)$$

$$y(f, \hat{\mu}, \hat{\alpha}) \geq 0 \quad \forall (f, \hat{\mu}, \hat{\alpha}) \in \Gamma^+ \quad (4.5b)$$

We claim that vector y in (4.5) can be chosen to be integer-valued and strictly positive. To see this, observe that if y is a feasible solution then so is any vector y' with $y'(f, \hat{\mu}, \hat{\alpha}) \in [Cy(f, \hat{\mu}, \hat{\alpha}), Cy(f, \hat{\mu}, \hat{\alpha}) + 1]$ for $(f, \hat{\mu}, \hat{\alpha}) \in \Gamma^+$, for some sufficiently large constant C (namely, $C > \frac{2}{\epsilon} \max_{f \in \Gamma, x \in \text{dom } f} |f(x)|$ where $\epsilon > 0$ is the minimum difference between the left-hand side and the right-hand side in (4.5a)).

Let us construct an instance \mathcal{I} with variables $V = \Delta^{(m)}$ and the function

$$f_{\mathcal{I}}(x) = \sum_{(f, \hat{\mu}, \hat{\alpha}) \in \Gamma^+, \hat{\alpha} = (\hat{\alpha}_1, \dots, \hat{\alpha}_n)} y(f, \hat{\mu}, \hat{\alpha}) f(x_{\hat{\alpha}_1}, \dots, x_{\hat{\alpha}_n}). \quad (4.6)$$

This can be viewed as a Γ -instance, if we simulate the multiplication of $y(f, \hat{\mu}, \hat{\alpha})$ and f by repeating the latter term $y(f, \hat{\mu}, \hat{\alpha})$ times.

Consider a mapping $g : \Delta^{(m)} \rightarrow D$. Since $V = \Delta^{(m)}$, such g is a valid labelling for the instance \mathcal{I} , so we can evaluate $f_{\mathcal{I}}(g)$. If $g \in \Omega^{(m)} \setminus \Omega_{\Gamma}^{(m)}$ then $f_{\mathcal{I}}(g) = \infty$ (by the

definition of $\Omega_\Gamma^{(m)}$, and if $g \in \Omega_\Gamma^{(m)}$ then $f_{\mathcal{I}}(g)$ equals the left-hand side of eq. (4.5a). Thus, eq. (4.5a) gives

$$\min_{g: V \rightarrow D} f_{\mathcal{I}}(g) > \sum_{(f, \hat{\mu}, \hat{\alpha}) \in \Gamma^+} y(f, \hat{\mu}, \hat{\alpha}) f(\hat{\mu}). \quad (4.7)$$

The left-hand side of eq. (4.7) is the optimal value of the instance \mathcal{I} . We claim that

$$\sum_{(f, \hat{\mu}, \hat{\alpha}) \in \Gamma^+} y(f, \hat{\mu}, \hat{\alpha}) f(\hat{\mu}) \geq \text{BLP}(\mathcal{I}) \quad (4.8)$$

and so $\min_g f_{\mathcal{I}}(g) > \text{BLP}(\mathcal{I})$, which contradicts the assumption that BLP solves Γ . To prove eq. (4.8), it suffices to specify a feasible vector of the BLP relaxation of \mathcal{I} (given by eq. (2.1)) whose value equals the left-hand side of (4.8). Such vector is constructed as follows: $\alpha_v = v$ for all $v \in V = \Delta^{(m)}$ and $\mu_t = \hat{\mu}$ for all $t = (f, \hat{\mu}, \hat{\alpha}) \in \Gamma^+$.

We showed that if Γ is finite then it admits a symmetric fractional polymorphism of arity m . Now suppose that Γ is infinite but countable: $\Gamma = \{f_1, f_2, \dots\}$. For any integer $r \geq 1$, denote by $\Gamma_r = \{f_1, \dots, f_r\}$. As we just showed, Γ_r admits some symmetric fractional polymorphism ω_r of arity m . The space of symmetric fractional polymorphisms is a compact subset of $\mathbb{R}^{|\Omega^{(m)}|}$, therefore the sequence $\omega_1, \omega_2, \dots$ has a limit vector ω . Using standard continuity arguments, we conclude that ω is a symmetric fractional polymorphism of Γ . This concludes the proof. \square

5. Constructing new fractional polymorphisms. In this section we introduce a generic procedure for constructing new fractional polymorphisms of a language Γ from existing ones. This procedure will be used in the proofs of Theorems 2 and 4, and Lemma 7.

We start with a motivating example to illustrate techniques that we will use.

EXAMPLE 6. *Let us consider all idempotent binary operations $D^2 \rightarrow D$ for the set of labels $D = \{0, 1\}$. In total, there are four such operations: $g_{00}, g_{01}, g_{10}, g_{11}$ where g_{ab} for $a, b \in D$ denotes the operation with*

$$g_{ab}(0, 0) = 0, \quad g_{ab}(0, 1) = a, \quad g_{ab}(1, 0) = b, \quad g_{ab}(1, 1) = 1.$$

The operations g_{01} and g_{10} are the two binary projections $e_1^{(2)}$ and $e_2^{(2)}$ to the first and second coordinate respectively. The operations g_{00} and g_{11} are the min and max operations with respect to the natural order on D .

Suppose that Γ admits a binary fractional polymorphism $\omega = \frac{1}{3}\chi_{g_{00}} + \frac{2}{3}\chi_{g_{01}}$ (so that one operation in $\text{supp}(\omega)$ is symmetric and the other is not), and we want to prove that Γ admits a symmetric binary fractional polymorphism. For a function $f \in \Gamma$ and labellings $x, y \in \text{dom } f$ we can write

$$\begin{aligned} \frac{f(x) + f(y)}{2} &= \frac{1}{2} \frac{f(x) + f(y)}{2} + \frac{1}{2} \frac{f(y) + f(x)}{2} \\ &\geq \frac{1}{2} \sum_{g \in \text{supp}(\omega)} \omega(g) f(g(x, y)) + \frac{1}{2} \sum_{g \in \text{supp}(\omega)} \omega(g) f(g(y, x)) \\ &= \frac{1}{3} f(g_{00}(x, y)) + \frac{2}{3} \frac{f(x) + f(y)}{2} \end{aligned} \quad (5.1)$$

This inequality means that vector $\rho_1 = \frac{1}{3}\chi_{g_{00}} + \frac{2}{3} \frac{\chi_{g_{01}} + \chi_{g_{10}}}{2}$ is a fractional polymorphism of Γ . This demonstrates how we can derive new fractional polymorphisms of a language from existing ones by taking superpositions.

To obtain a symmetric fractional polymorphism of Γ , we can now use two strategies:

- (i) Cancel terms in (5.1), obtaining inequality $\frac{1}{3} \frac{f(x)+f(y)}{2} \geq \frac{1}{3} f(g_{00}(x, y))$. This inequality means that vector $\chi_{g_{00}}$ is a fractional polymorphism of Γ .
- (ii) Take the last term $\frac{2}{3} \frac{f(x)+f(y)}{2}$ in (5.1) and apply ω to it again. This gives a new inequality corresponding to a fractional polymorphism $\rho_2 = \frac{5}{9} \chi_{g_{00}} + \frac{4}{9} \frac{\chi_{g_{01}} + \chi_{g_{10}}}{2}$. By repeating this process we obtain a sequence of vectors $\rho_1, \rho_2, \rho_3, \dots$ that are fractional polymorphisms of Γ . The weight of operation g_{00} in ρ_i tends to 1 as i tends to infinity; thus, by taking the limit we can prove that vector $\chi_{g_{00}}$ is a fractional polymorphism of Γ .

Observe that in the construction in Example 6 operations g_{01} and g_{10} always had the same weight in ρ_i . Therefore, we were working with a subset of all possible fractional polymorphisms. This example motivates definitions given below.

5.1. Generalised fractional polymorphisms. The construction of new fractional polymorphisms is based on the idea of grouping operations in $\mathcal{O}^{(m)}$ together into what we will call *collections* and working with fractional operations that assign the same weight to every operation in a collection. We will consider two types of collections: *ordered* and *unordered*. Ordered collections are finite sequences of operations from $\mathcal{O}^{(m)}$ and unordered collections are subsets of $\mathcal{O}^{(m)}$.

Let \mathbb{G} be a fixed set of collections. We will always assume that all collections in \mathbb{G} are of the same type, i.e., either ordered or unordered. (In Example 6 above we would use $\mathbb{G} = \{\{g_{00}\}, \{g_{11}\}, \{g_{01}, g_{10}\}\}$.) For a collection $\mathbf{g} \in \mathbb{G}$, we let $|\mathbf{g}|$ denote its *size*, i.e., the cardinality of the set, or the length of the sequence, depending on its type. We write $\sum_{g \in \mathbf{g}}$ to denote a sum over all components of the (ordered or unordered) collection \mathbf{g} ; one has e.g. $\sum_{g \in \mathbf{g}} 1 = |\mathbf{g}|$.

For a given \mathbb{G} , we define a *generalised fractional polymorphism* ρ of a language Γ as a probability distribution over \mathbb{G} such that, for every cost function $f \in \Gamma$,

$$\sum_{\mathbf{g} \in \mathbb{G}} \rho(\mathbf{g}) \sum_{g \in \mathbf{g}} \frac{1}{|\mathbf{g}|} f(g(x^1, \dots, x^m)) \leq f^m(x^1, \dots, x^m), \quad \forall x^1, \dots, x^m \in \text{dom } f. \quad (5.2)$$

To simplify notation, we will write (5.2) as

$$\sum_{\mathbf{g} \in \mathbb{G}} \rho(\mathbf{g}) f^{|\mathbf{g}|}(\mathbf{g}(x^1, \dots, x^m)) \leq f^m(x^1, \dots, x^m), \quad \forall x^1, \dots, x^m \in \text{dom } f, \quad (5.3)$$

where, for an unordered collection $\mathbf{g} = \{g_1, \dots, g_k\}$ of m -ary operations, the application $\mathbf{g}(x^1, \dots, x^m)$ denotes the set of labellings $\{g_1(x^1, \dots, x^m), \dots, g_k(x^1, \dots, x^m)\}$, and we define $f^k(\{y^1, \dots, y^k\}) = f^k(y^1, \dots, y^k)$ for labellings $y^1, \dots, y^k \in D^n$. Similarly, for an ordered collection $\mathbf{g} = (g_1, \dots, g_k)$ of m -ary operations, the application $\mathbf{g}(x^1, \dots, x^m)$ denotes the sequence of labellings $(g_1(x^1, \dots, x^m), \dots, g_k(x^1, \dots, x^m))$.

As for fractional operations, we define the $\text{supp}(\rho)$ as the set of collections \mathbf{g} for which $\rho(\mathbf{g}) > 0$, and the vector $\chi_{\mathbf{g}}$ as the vector that assigns weight 1 to the collection \mathbf{g} and 0 to all other collections.

Note that ρ is a generalised fractional polymorphism of Γ if and only if $\omega = \sum_{\mathbf{g} \in \mathbb{G}} \rho(\mathbf{g}) \sum_{g \in \mathbf{g}} \frac{1}{|\mathbf{g}|} \chi_g$ is an m -ary fractional polymorphism of Γ .

Terminology for ordered collections. Observe that an ordered collection $\mathbf{g} = (g_1, \dots, g_k)$ is a mapping $D^m \rightarrow D^k$. Denote the set of such mappings by $\mathcal{O}^{(m \rightarrow k)}$. If \mathbb{G} is a subset of $\mathcal{O}^{(m \rightarrow k)}$, then a generalised fractional polymorphism ρ over \mathbb{G} will be called a *generalised fractional polymorphism of arity $m \rightarrow k$* . We can identify fractional polymorphisms of arity m with generalised fractional polymorphisms of arity $m \rightarrow 1$.

5.2. Constructing generalised fractional polymorphisms. Our goal will be to construct a generalised fractional polymorphism ρ such that all collections $\mathbf{g} \in \text{supp}(\rho)$ satisfy some desired property. Let $\mathbb{G}^* \subseteq \mathbb{G}$ be the set of such “good” collections. We consider an *expansion operator* Exp that takes a collection $\mathbf{g} \in \mathbb{G}$ and produces a probability distribution ρ over \mathbb{G} . We say that Exp is *valid* for a language Γ if, for any $f \in \Gamma$ and any $\mathbf{g} \in \mathbb{G}$, the probability distribution $\rho = \text{Exp}(\mathbf{g})$ satisfies

$$\sum_{\mathbf{h} \in \mathbb{G}} \rho(\mathbf{h}) f^{|\mathbf{h}|}(\mathbf{h}(x^1, \dots, x^m)) \leq f^{|\mathbf{g}|}(\mathbf{g}(x^1, \dots, x^m)), \quad \forall x^1, \dots, x^m \in \text{dom } f. \quad (5.4)$$

We say that the operator Exp is *non-vanishing* (with respect to the pair $(\mathbb{G}, \mathbb{G}^*)$) if, for any $\mathbf{g} \in \mathbb{G}$, there exists a sequence of collections $\mathbf{g}_0, \mathbf{g}_1, \dots, \mathbf{g}_r$ such that $\mathbf{g}_0 = \mathbf{g}$, $\mathbf{g}_{i+1} \in \text{supp}(\text{Exp}(\mathbf{g}_i))$ for $i = 0, \dots, r-1$, and $\mathbf{g}_r \in \mathbb{G}^*$.

The main result of this section is the following.

LEMMA 10 (“Expansion Lemma”). *Let Exp be an expansion operator which is valid for the language Γ and non-vanishing with respect to $(\mathbb{G}, \mathbb{G}^*)$. If Γ admits a generalised fractional polymorphism ρ with $\text{supp}(\rho) \subseteq \mathbb{G}$, then it also admits a generalised fractional polymorphism ρ^* with $\text{supp}(\rho^*) \subseteq \mathbb{G}^*$.*

The example below demonstrates how this lemma can be used.

EXAMPLE 6 (revisited). *Recall that in this example we have a language Γ over $D = \{0, 1\}$ that admits a fractional polymorphism $\omega = \frac{1}{3}\chi_{g_{00}} + \frac{2}{3}\chi_{g_{01}}$. Let us show how we can use the Expansion Lemma to derive the fact that Γ admits a symmetric fractional polymorphism.*

Let $\mathbb{G} = \{\{g_{00}\}, \{g_{11}\}, \{g_{01}, g_{10}\}\}$ and $\mathbb{G}^ = \{\{g_{00}\}, \{g_{11}\}\}$. Note that vector $\rho = \chi_{\{g_{01}, g_{10}\}}$ is a generalised fractional polymorphism of Γ . Define expansion operator Exp as follows:*

$$\begin{aligned} \text{Exp}(\{g_{00}\}) &= \sum_{g \in \text{supp}(\omega)} \omega(g) \chi_{\{g[g_{00}, g_{00}]\}} = \chi_{\{g_{00}\}} \\ \text{Exp}(\{g_{11}\}) &= \sum_{g \in \text{supp}(\omega)} \omega(g) \chi_{\{g[g_{11}, g_{11}]\}} = \chi_{\{g_{11}\}} \\ \text{Exp}(\{g_{01}, g_{10}\}) &= \sum_{g \in \text{supp}(\omega)} \omega(g) \chi_{\{g[g_{01}, g_{10}], g[g_{10}, g_{01}]\}} = \frac{1}{3}\chi_{\{g_{00}\}} + \frac{2}{3}\chi_{\{g_{01}, g_{10}\}} \end{aligned}$$

It can be verified Exp is valid for Γ and non-vanishing with respect to $(\mathbb{G}, \mathbb{G}^)$; we leave this to the reader. By the Expansion Lemma, Γ admits a generalised fractional polymorphism ρ^* with $\text{supp}(\rho^*) \subseteq \mathbb{G}^*$, i.e. $\rho^* = \rho_{00}^* \chi_{\{g_{00}\}} + \rho_{11}^* \chi_{\{g_{11}\}}$ for some $\rho_{00}^*, \rho_{11}^* \geq 0$ with $\rho_{00}^* + \rho_{11}^* = 1$. This means that vector $\rho_{00}^* \chi_{g_{00}} + \rho_{11}^* \chi_{g_{11}}$ is a (symmetric) fractional polymorphism of Γ .*

Note that in the original Example 6 we claimed a stronger property, namely that Γ admits a fractional polymorphism $\chi_{g_{00}}$. This can be established by taking $\mathbb{G} = \{\{g_{00}\}, \{g_{01}, g_{10}\}\}$ and $\mathbb{G}^ = \{\{g_{00}\}\}$; the expansion operator for these collections is defined as above.*

The lemma will be used for constructing the desired fractional polymorphisms in Theorem 2 and Lemma 7; their proofs consist of exhibiting an appropriate pair $(\mathbb{G}, \mathbb{G}^*)$ and an expansion operator Exp . We will also exploit the lemma in the proof of Theorem 4. However, on its own it will not be enough, and we will need an additional method for constructing symmetric fractional polymorphisms. Considering Example 5, this should not come as a surprise: Theorem 4 holds only for finite-valued languages, while Lemma 10 is valid for all general-valued languages.

We will now give two proofs of Lemma 10. The first one is constructive: it shows how to obtain ρ^* from ρ using a finite number of steps. The second one is shorter but non-constructive. Roughly speaking, the constructive proof uses strategy (i) from Example 6, while the non-constructive one uses strategy (ii).

5.3. Constructive proof of the Expansion Lemma. In this subsection, we give a constructive proof of the Expansion Lemma based on a node-weighted tree generated by repeated applications of the expansion operator.

Proof. (of Lemma 10) The proof goes via an explicit construction of a node-weighted tree. Each node of the tree contains a collection $\mathbf{g} \in \mathbb{G}$, and we will use \mathbf{g} to denote both the node and the collection of operations it contains. We say that two nodes are *equal as collections* if the collections they contain are the same. Each node also carries a (strictly) positive weight, denoted by $w(\mathbf{g})$. We say that a node \mathbf{g} is *covered (by \mathbf{h})*, and that \mathbf{h} is a *covering node (of \mathbf{g})* if \mathbf{g} is a descendant of \mathbf{h} and \mathbf{g} and \mathbf{h} are equal as collections. We say that \mathbf{h} is a *minimal covering node* if no descendant of \mathbf{h} is a covering node.

The root of the tree will be denoted by \mathbf{p} . If the collections in \mathbb{G} are unordered, then we let $\mathbf{p} = \{e_1^{(m)}, \dots, e_m^{(m)}\}$ be the set of m -ary projections. If the collections are ordered, then we let $\mathbf{p} = (e_1^{(m)}, \dots, e_m^{(m)})$. If \mathbf{p} is not in \mathbb{G} , then we can augment \mathbb{G} with \mathbf{p} and define $\text{Exp}(\mathbf{p})$ to be ρ . In both cases, since Γ admits ρ , the augmented expansion operator remains non-vanishing and valid for Γ .

The construction is performed in two steps. In the first step, the *expansion*, a tree with root \mathbf{p} is constructed. At the end of this step, the tree will have leaves that are either in \mathbb{G}^* or that are covered. From the construction, it will immediately follow that the leaves induce a generalised fractional polymorphism of Γ . In the second step, the *pruning*, certain parts of the tree will be cut down to remove all leaves that are not in \mathbb{G}^* . We then prove that the set of leaves remaining after the pruning step still induces a generalised fractional polymorphism of Γ .

The expansion step is carried out as follows.

- While there exists a leaf \mathbf{g} in the tree that is in $\mathbb{G} \setminus \mathbb{G}^*$ and not covered, add the set $\text{supp}(\text{Exp}(\mathbf{g}))$ as children to \mathbf{g} with the weight of each added node \mathbf{h} given by $w(\mathbf{g}) \cdot \text{Exp}(\mathbf{g})(\mathbf{h})$.

First, we argue that the expansion step terminates after a finite number of applications of the expansion operator. Assume to the contrary that the expansion generates an infinite tree. Since there is a finite number of m -ary operations, $|\mathbb{G}|$ is finite, and hence it follows that the tree has an infinite path $\mathbf{g}_0, \mathbf{g}_1, \dots$, descending from the root. Therefore, there exist $i < j$ such that $\mathbf{g}_i = \mathbf{g}_j$ as collections. If $\mathbf{g}_j \in \mathbb{G}^*$, then \mathbf{g}_i would never have been expanded, so we may assume that $\mathbf{g}_j \notin \mathbb{G}^*$. But then \mathbf{g}_j is covered by \mathbf{g}_i , hence \mathbf{g}_j would never have been expanded. We have reached a contradiction and it follows that the tree generated by the expansion is finite.

Let T be the tree generated by the expansion step. For a node \mathbf{g} in T , let $L(\mathbf{g})$ be the set of leaves of the sub-tree rooted at \mathbf{g} and define the vector $\nu_{\mathbf{g}} \in \mathbb{R}_{\geq 0}^{\mathbb{G}}$ by $\nu_{\mathbf{g}} = \sum_{\mathbf{h} \in L(\mathbf{g})} (w(\mathbf{h})/w(\mathbf{g})) \chi_{\mathbf{h}}$. We claim that the following three properties are satisfied:

- (a) For every node \mathbf{g} in T , the vector $\nu_{\mathbf{g}}$ is a probability distribution on \mathbb{G} such that for every function $f \in \Gamma$, and all tuples $x^1, \dots, x^m \in \text{dom } f$,

$$\sum_{\mathbf{h} \in \mathbb{G}} \nu_{\mathbf{g}}(\mathbf{h}) f^{|\mathbf{h}|}(\mathbf{h}(x^1, \dots, x^m)) \leq f^{|\mathbf{g}|}(\mathbf{g}(x^1, \dots, x^m)). \quad (5.5)$$

- (b) Every leaf in T is either a member of \mathbb{G}^* or covered.

(c) Every sub-tree of T contains a leaf that is not covered by a node in that sub-tree.

Property (a) follows by repeated application of (5.4), with special care taken to the case $\nu_{\mathbf{p}}$, where the fact that ρ satisfies (5.3) is also used. Property (b) holds trivially when the expansion step terminates. To see that (c) holds, assume to the contrary that, after the expansion step, there is a sub-tree rooted at \mathbf{g} for which every leaf is in $\mathbb{G} \setminus \mathbb{G}^*$. Pick any node \mathbf{g}_0 in this sub-tree, and for $i \geq 0$, arbitrarily pick $\mathbf{g}_{i+1} \in \text{supp}(\text{Exp}(\mathbf{g}_i))$. Then, each \mathbf{g}_i contains a collection that already exists in the sub-tree, hence the sequence $\mathbf{g}_0, \mathbf{g}_1, \dots$ never encounters a collection in \mathbb{G}^* . This is a contradiction since Exp is assumed to be non-vanishing. Hence, (c) holds after the expansion step.

The pruning step is carried out as follows.

- While there exists a covered leaf in the tree, pick a minimal covering node \mathbf{g} and let $T_{\mathbf{g}}$ be the sub-tree rooted at \mathbf{g} . Write $\nu_{\mathbf{g}}$ as $\nu_{\mathbf{g}} = (1 - \kappa) \cdot \chi_{\mathbf{g}} + \nu_{\perp}$, where $1 - \kappa = \nu_{\mathbf{g}}(\mathbf{g})$ so that $\nu_{\perp}(\mathbf{g}) = 0$. Remove all nodes below \mathbf{g} from the tree and, for each collection $\mathbf{h} \in \mathbb{G}$ such that $\nu_{\perp}(\mathbf{h}) > 0$, add a new leaf \mathbf{h}' as a child to \mathbf{g} containing the collection \mathbf{h} and with weight $w(\mathbf{h}') = w(\mathbf{g}) \frac{1}{\kappa} \nu_{\perp}(\mathbf{h})$.

We refer to each choice of a minimal covering node \mathbf{g} and the subsequent restructuring of $T_{\mathbf{g}}$ as a *round* of pruning. Below, we prove that the properties (a–c) are invariants that hold before and after each round of pruning. This has the following consequences. Since (c) holds before each round, some leaf \mathbf{h} of \mathbf{g} is different from \mathbf{g} as a collection, and since $w(\mathbf{h})$ is strictly positive, $\kappa > 0$. Therefore the new weights in the pruning step are defined. Each round of pruning decreases the size of the tree by at least one, so the pruning step eventually terminates. After the pruning step has terminated, let ρ^* be $\nu_{\mathbf{p}}$. By (a), ρ^* is a generalised fractional polymorphism of Γ , and by (b), every leaf must contain a collection in \mathbb{G}^* , so $\text{supp}(\rho^*) \subseteq \mathbb{G}^*$.

We finish the proof by showing that (a–c) hold after a round of pruning that picks a minimal covering node \mathbf{g} , assuming that they held before the round. By (a) and noting that $\sum_{\mathbf{h} \in \mathbb{G}} \chi_{\mathbf{g}}(\mathbf{h}) f^{|\mathbf{h}|}(\mathbf{h}(x^1, \dots, x^m)) = f^{|\mathbf{g}|}(\mathbf{g}(x^1, \dots, x^m))$, we have

$$\begin{aligned} \sum_{\mathbf{h} \in \mathbb{G}} \nu_{\mathbf{g}}(\mathbf{h}) f^{|\mathbf{h}|}(\mathbf{h}(x^1, \dots, x^m)) &= \sum_{\mathbf{h} \in \mathbb{G}} ((1 - \kappa) \cdot \chi_{\mathbf{g}} + \nu_{\perp})(\mathbf{h}) f^{|\mathbf{h}|}(\mathbf{h}(x^1, \dots, x^m)) \\ &\geq \sum_{\mathbf{h} \in \mathbb{G}} ((1 - \kappa) \cdot \nu_{\mathbf{g}} + \nu_{\perp})(\mathbf{h}) f^{|\mathbf{h}|}(\mathbf{h}(x^1, \dots, x^m)), \end{aligned} \quad (5.6)$$

for all $f \in \Gamma$ and $x^1, \dots, x^m \in \text{dom } f$. Since $\kappa > 0$, inequality (5.6) is equivalent to

$$\sum_{\mathbf{h} \in \mathbb{G}} \nu_{\mathbf{g}}(\mathbf{h}) f^{|\mathbf{h}|}(\mathbf{h}(x^1, \dots, x^m)) \geq \sum_{\mathbf{h} \in \mathbb{G}} \frac{1}{\kappa} \nu_{\perp}(\mathbf{h}) f^{|\mathbf{h}|}(\mathbf{h}(x^1, \dots, x^m)). \quad (5.7)$$

The round of pruning only affects the vector $\nu_{\mathbf{g}'}$ for nodes \mathbf{g}' that lie on the path from the root to \mathbf{g} . Let \mathbf{g}' be such a node and let $\nu'_{\mathbf{g}'}$ be the altered function after the round. Let $C = w(\mathbf{g})/w(\mathbf{g}')$. Then, $\nu'_{\mathbf{g}'} = \nu_{\mathbf{g}'} - C\nu_{\mathbf{g}} + C\frac{1}{\kappa}\nu_{\perp}$ and $\nu'_{\mathbf{g}'}$ can easily be verified to be a probability distribution on \mathbb{G} , and

$$\begin{aligned}
f^{|\mathbf{g}'|}(\mathbf{g}'(x^1, \dots, x^m)) &\geq \sum_{\mathbf{h} \in \mathbb{G}} (\nu_{\mathbf{g}'} - C\nu_{\mathbf{g}} + C\nu_{\mathbf{g}})(\mathbf{h}) f^{|\mathbf{h}|}(\mathbf{h}(x^1, \dots, x^m)) \\
&\geq \sum_{\mathbf{h} \in \mathbb{G}} (\nu_{\mathbf{g}'} - C\nu_{\mathbf{g}} + C\frac{1}{\kappa}\nu_{\perp})(\mathbf{h}) f^{|\mathbf{h}|}(\mathbf{h}(x^1, \dots, x^m)) \\
&= \sum_{\mathbf{h} \in \mathbb{G}} \nu'_{\mathbf{g}'}(\mathbf{h}) f^{|\mathbf{h}|}(\mathbf{h}(x^1, \dots, x^m)), \tag{5.8}
\end{aligned}$$

so (a) holds after the round.

Before the round of pruning, using (b) and the fact that \mathbf{g} is a minimal covering node, every leaf in $T_{\mathbf{g}}$ is either equal to \mathbf{g} as collections, is a member of \mathbb{G}^* , or is covered by a node above \mathbf{g} in the tree. Therefore, every new child \mathbf{h}' added to \mathbf{g} in the pruning is either a member of \mathbb{G}^* or is still covered by some node above \mathbf{g} , so (b) holds after the round.

The round of pruning only affects the leaves of the sub-trees $T_{\mathbf{g}'}$ that contain \mathbf{g} . Before the round, by (c), the sub-tree $T_{\mathbf{g}'}$ either contains a leaf in \mathbb{G}^* or a leaf that is covered by a node above \mathbf{g}' in the tree. If such a leaf \mathbf{h} is also a leaf of $T_{\mathbf{g}}$, and therefore potentially altered in the round of pruning, then \mathbf{h} is different from \mathbf{g} as collections, hence \mathbf{g} will have a new child \mathbf{h}' with the same property. It follows that (c) holds after the round. \square

5.4. Non-constructive proof of the Expansion Lemma. We now give a non-constructive proof of the Expansion Lemma.

Proof. (of Lemma 10) Let Ω be the set of generalised fractional polymorphisms ρ of Γ with $\text{supp}(\rho) \subseteq \mathbb{G}$; it is non-empty by the assumption of Lemma 10. Let us pick $\rho^* \in \Omega$ with the maximum value of $\rho^*(\mathbb{G}^*) = \sum_{\mathbf{g} \in \mathbb{G}^*} \rho(\mathbf{g})$. (Clearly, Ω is a compact set which is a subset of $\mathbb{R}^{|\mathbb{G}|}$, so the maximum is attained by some vector in Ω). We claim that $\text{supp}(\rho^*) \subseteq \mathbb{G}^*$. Indeed, suppose that $\mathbf{g}_0 \notin \mathbb{G}^*$ for some $\mathbf{g}_0 \in \text{supp}(\rho^*)$. Since Exp is non-vanishing, there exists a sequence $\mathbf{g}_0, \mathbf{g}_1, \dots, \mathbf{g}_r$ such that $\mathbf{g}_{i+1} \in \text{supp}(\text{Exp}(\mathbf{g}_i))$ for $i = 0, \dots, r-1$, $\mathbf{g}_0, \dots, \mathbf{g}_{r-1} \notin \mathbb{G}^*$, and $\mathbf{g}_r \in \mathbb{G}^*$. Define a sequence of generalised fractional polymorphisms $\rho_0 = \rho^*, \rho_1, \dots, \rho_r$ as follows:

$$\rho_{i+1} = \rho_i + \rho_i(\mathbf{g}_i) \cdot (-\chi_{\mathbf{g}_i} + \nu_i), \quad \nu_i = \text{Exp}(\mathbf{g}_i)$$

for $i = 0, \dots, r-1$. We can prove by induction on i that $\rho_i \in \Omega$, i.e. ρ_i is a generalised fractional polymorphism of Γ . Indeed, for any $f \in \Gamma$, and $x^1, \dots, x^m \in \text{dom } f$, we have

$$\sum_{\mathbf{h} \in \mathbb{G}} (-\chi_{\mathbf{g}_i} + \nu_i)(\mathbf{h}) f^{|\mathbf{h}|}(\mathbf{h}(x^1, \dots, x^m)) \leq 0$$

where we used eq. (5.4) and the fact that $f^{|\mathbf{g}_i|}(\mathbf{g}_i(x^1, \dots, x^m))$ is finite (since $\mathbf{g}_i \in \text{supp}(\rho_i)$ and ρ_i is a generalised fractional polymorphism of f). Therefore,

$$\sum_{\mathbf{h} \in \mathbb{G}} \rho_{i+1}(\mathbf{h}) f^{|\mathbf{h}|}(\mathbf{h}(x^1, \dots, x^m)) \leq \sum_{\mathbf{h} \in \mathbb{G}} \rho_i(\mathbf{h}) f^{|\mathbf{h}|}(\mathbf{h}(x^1, \dots, x^m)) \leq f^m(\mathbf{g}(x^1, \dots, x^m))$$

(the last inequality is by the induction hypothesis). We proved that $\rho_i \in \Omega$ for all i .

By construction, $\rho^*(\mathbb{G}^*) = \rho_0(\mathbb{G}^*) \leq \rho_1(\mathbb{G}^*) \leq \dots \leq \rho_{r-1}(\mathbb{G}^*) < \rho_r(\mathbb{G}^*)$. This contradicts the choice of ρ^* . \square

6. Second characterisation of general-valued languages. In this section, we use the Expansion Lemma from Section 5 to prove Theorem 2.

THEOREM 2 (restated). *Suppose that, for every $n \geq 2$, Γ admits a fractional polymorphism ω_n such that $\text{supp}(\omega_n)$ generates a symmetric n -ary operation. Then, Γ admits a symmetric fractional polymorphism of every arity $m \geq 2$.*

Proof. The proof is an application of Lemma 10. Fix some arbitrary arity $m \geq 2$. Let \sim denote the following equivalence relation on the set $\mathcal{O}^{(m)}$ of m -ary operations on D :

$$g \sim g' \Leftrightarrow \exists \pi \in S_m : g(x_1, \dots, x_m) = g'(x_{\pi(1)}, \dots, x_{\pi(m)}).$$

Let \mathbb{G} be the set of equivalence classes of the relation \sim and let \mathbb{G}^* be the set of all equivalence classes $\mathbf{g} \in \mathbb{G}$ for which $|\mathbf{g}| = 1$, i.e., the set of equivalence classes containing a single symmetric operation. We say that a fractional operation ν is *weight-symmetric* if $\nu(g) = \nu(g')$ whenever $g \sim g'$. A weight-symmetric fractional operation ν induces a probability distribution ρ on \mathbb{G} : $\rho(\mathbf{g}) = \nu(g)/|\mathbf{g}|$, where g is any of the operations in \mathbf{g} .

We now define the expansion operator Exp by giving its result when applied to an arbitrary $\mathbf{g} \in \mathbb{G}$. Let $n = |\mathbf{g}|$ and let ω be a k -ary fractional polymorphism of Γ such that $\text{supp}(\omega)$ generates a symmetric n -ary operation.

Define a sequence of m -ary weight-symmetric fractional operations ν_0, ν_1, \dots , each with $\|\nu_i\|_1 = 1$, as follows. Let $\nu_0 = \sum_{g \in \mathbf{g}} \frac{1}{|\mathbf{g}|} \chi_g$. For $i \geq 1$, assume that ν_{i-1} has been defined. Let $l_{i-1} = \min\{\nu_{i-1}(g) \mid g \in \text{supp}(\nu_{i-1})\}$ be the minimum weight of an operation in the support of ν_{i-1} . The fractional operation ν_i is obtained by subtracting from ν_{i-1} an equal weight from each operation in $\text{supp}(\nu_{i-1})$ and adding back this weight as superpositions of ω by all possible choices of operations in $\text{supp}(\nu_{i-1})$. The amount subtracted from each operation is $\frac{1}{2}l_{i-1}$. This implies that every collection in $\text{supp}(\nu_{i-1})$ is also in $\text{supp}(\nu_i)$.

Formally ν_i is defined as follows:

$$\nu_i = \nu_{i-1} - \frac{l_{i-1}}{2} \chi_{i-1} + \frac{l_{i-1}}{2} \eta_{i-1},$$

where $\chi_{i-1} = \frac{1}{|\text{supp}(\nu_{i-1})|} \sum_{g \in \text{supp}(\nu_{i-1})} \chi_g$ and, with $K = |\text{supp}(\nu_{i-1})|^k$,

$$\eta_{i-1} = \frac{1}{K} \sum_{g_1, \dots, g_k \in \text{supp}(\nu_{i-1})} \omega[g_1, \dots, g_k].$$

By definition, $\nu \geq 0$ and $\|\nu_i\|_1 = \|\nu_{i-1}\|_1 = 1$. To see that ν_i is weight-symmetric, it suffices to verify that η_{i-1} is weight-symmetric. Let g be any m -ary operation of the form $g = h[g_1, \dots, g_k]$ and let $g \sim g'$. Let $\pi \in S_m$ be such that $g(x_1, \dots, x_m) = g'(x_{\pi(1)}, \dots, x_{\pi(m)})$ and define $g'_j(x_1, \dots, x_m) = g_j(x_{\pi(1)}, \dots, x_{\pi(m)})$ for $1 \leq j \leq k$. Since ν_{i-1} is weight-symmetric, it follows that $g_i \in \text{supp}(\nu_{i-1})$ if and only if $g'_i \in \text{supp}(\nu_{i-1})$. Therefore the terms $\omega(h)h[g_1, \dots, g_k]$ in η_{i-1} such that $g = h[g_1, \dots, g_k]$ are in bijection with the terms $\omega(h)h[g'_1, \dots, g'_k]$ such that $g' = h[g'_1, \dots, g'_k]$. So the fractional operation η_{i-1} assigns the same weight to g and g' .

The assumption that $\text{supp}(\omega)$ generates a symmetric n -ary operation t means that t can be obtained by a finite number of superpositions of operations from $\text{supp}(\omega)$ and the set of all projections. Formally, define $\mathcal{A}_0 = \{e_j^{(i)} \mid 1 \leq j \leq i\}$ to be the set of

all projections and, for $j \geq 1$, define $\mathcal{A}_j = \{g[h_1, \dots, h_q] \mid g \in \text{supp}(\omega), h_1, \dots, h_q \in \mathcal{A}_{j'}, j' < j\}$. Then, $t \in \mathcal{A}_d$, for some $d \geq 1$. Fix any such value of d and define $\text{Exp}(\mathbf{g})$ to be the probability distribution on \mathbb{G} induced by ν_d .

We now show that this operator is non-vanishing. Assume that $\mathbf{g} = \{g_1, \dots, g_n\}$. Since $\text{supp}(\nu_0) = \{g_1, \dots, g_n\}$ and using the fact that $\text{supp}(\nu_i)$ contains all superpositions $g[h_1, \dots, h_q]$, $g \in \text{supp}(\omega)$, $h_1, \dots, h_q \in \text{supp}(\nu_{i-1})$, it follows by induction that $t[g_1, \dots, g_n] \in \text{supp}(\nu_d)$. Note that the operation $t[g_1, \dots, g_n]$ is symmetric since for all $\pi \in S_m$, there is a permutation $\pi' \in S_n$ such that $t[g_1, \dots, g_n](x_{\pi(1)}, \dots, x_{\pi(m)}) = t[g_{\pi'(1)}, \dots, g_{\pi'(n)}](x_1, \dots, x_m) = t[g_1, \dots, g_n](x_1, \dots, x_m)$. Hence, $\text{Exp}(\mathbf{g})$ assigns non-zero probability to $\{t[g_1, \dots, g_n]\}$ and since $\{t[g_1, \dots, g_n]\} \in \mathbb{G}^*$, it follows that Exp is non-vanishing.

It remains to show that Exp is valid for Γ , i.e., that (5.4) is satisfied. We claim that for each $i \geq 1$, we have

$$\sum_{g \in \mathcal{O}^{(m)}} \nu_i(g) f(g(x^1, \dots, x^m)) \leq \sum_{g \in \mathcal{O}^{(m)}} \nu_{i-1}(g) f(g(x^1, \dots, x^m)), \quad (6.1)$$

for all $f \in \Gamma$ and $x^1, \dots, x^m \in \text{dom } f$. The inequality (5.4) then follows by induction on i , noting that $\sum_{g \in \mathcal{O}^{(m)}} \nu_0(g) f(g(x^1, \dots, x^m)) = \sum_{g \in \mathbf{g}} \frac{1}{|\mathbf{g}|} f(g(x^1, \dots, x^m)) = f^{|\mathbf{g}|}(\mathbf{g}(x^1, \dots, x^m))$. To see why (6.1) holds, compare the last two terms in the definition of ν_i :

$$\begin{aligned} \sum_{g \in \mathcal{O}^{(m)}} \chi_{i-1}(g) f(g(x^1, \dots, x^m)) &= \frac{1}{K} \sum_{g_1, \dots, g_k \in \text{supp}(\nu_{i-1})} \frac{1}{k} \sum_{i=1}^k f(g_i(x^1, \dots, x^m)) \\ &\geq \frac{1}{K} \sum_{g_1, \dots, g_k \in \text{supp}(\nu_{i-1})} \sum_{h \in \mathcal{O}^{(m)}} \omega(h) f(h[g_1, \dots, g_k](x^1, \dots, x^m)) \\ &= \sum_{h' \in \mathcal{O}^{(m)}} \eta_{i-1}(h') f(h'(x^1, \dots, x^m)). \end{aligned}$$

Hence, Exp is valid, so Lemma 10 is applicable and shows that Γ admits a generalised fractional polymorphism ρ^* with support on singleton sets, each containing a symmetric m -ary operation. Therefore, Γ admits the symmetric m -ary fractional polymorphism $\sum_{\{g\} \in \mathbb{G}^*} \rho^*(\{g\}) \chi_g$. \square

7. Imposing idempotency. In this section we prove Lemma 7 which was used to find optimal solutions in Section 3.3. The lemma states that for a symmetric fractional polymorphism, we can impose idempotency on a sub-domain D' of D while simultaneously ensuring that there is an optimal solution with labels restricted to D' . The proof uses the Expansion Lemma.

LEMMA 7 (restated). *There exists a subset $D' \subseteq D$ such that if Γ admits an m -ary symmetric fractional polymorphism, then it admits an m -ary symmetric fractional polymorphism ω such that, for all $g \in \text{supp}(\omega)$,*

1. $g(x, x, \dots, x) \in D'$ for all $x \in D$;
2. $g(x, x, \dots, x) = x$ for all $x \in D'$.

Proof. Every language Γ admits the fractional polymorphism that assigns probability 1 to the unary identity operation on D . Furthermore, assuming that Γ admits two unary fractional polymorphisms ν_1 and ν_2 , Γ also admits $\nu' = \frac{1}{2}(\nu_1 + \nu_2)$ with $\text{supp}(\nu') = \text{supp}(\nu_1) \cup \text{supp}(\nu_2)$. Therefore, we can let ν be a unary fractional polymorphism of Γ with inclusion-maximal support. Let $h \in \text{supp}(\nu)$ be such that $|h(D)| = \min\{|g(D)| \mid g \in \text{supp}(\nu)\}$ and define $D' = h(D)$.

Let $\mathbb{G} = \{\{g\} \mid g \in \mathcal{O}_{\text{sym}}^{(m)}\}$ and let \mathbb{G}^* be the operations in \mathbb{G} that additionally satisfy (1) and (2). The expansion operator Exp is defined as follows: $\text{Exp}(\{g\})$ assigns probability $\sum_{h' \in \text{supp}(\nu), g' = h' \circ g} \nu(h')$ to the set $\{g'\}$. It is easy to see that Exp is valid and we show below that it is non-vanishing. Therefore, Lemma 10 is applicable with ρ taken to be an m -ary symmetric fractional polymorphism of Γ . Consequently, Γ admits $\omega = \sum_{\{g\} \in \mathbb{G}^*} \rho^*(\{g\}) \chi_g$.

We finish the proof by showing that Exp is non-vanishing. It is easy to see that Γ admits the unary fractional polymorphism $\mu = \sum_{h_1 \in \text{supp}(\nu)} \nu(h_1) \sum_{h_2 \in \text{supp}(\nu)} \nu(h_2) \chi_{h_1 \circ h_2}$. Since ν is inclusion-maximal, it follows that $h_1 \circ h_2 \in \text{supp}(\mu) \subseteq \text{supp}(\nu)$, so $\text{supp}(\nu)$ forms a monoid under composition.

Define G as $\{g|_{D'} \mid g \in \text{supp}(\nu), g(D) = D'\}$. Then, G is a set of permutations of D' that contains the identity. Let $g'_1, g'_2 \in G$ be two permutations in this set and let $g_1, g_2 \in \text{supp}(\nu)$ be such that $g_1(D) = g_2(D) = D'$ and $g'_i = g_i|_{D'}$. Since $\text{supp}(\nu)$ forms a monoid under composition, $g_1 \circ g_2 \in \text{supp}(\nu)$. Therefore, $g'_1 \circ g'_2 = g_1|_{D'} \circ g_2|_{D'} = (g_1 \circ g_2)|_{D'} \in G$, so G forms a group under composition.

Let $\{g\} \in \mathbb{G}$. By the *diagonal* of an operation f we mean the unary operation $x \mapsto f(x, \dots, x)$. Note that the diagonal of $h \circ g$ acts as a permutation on D' . This permutation has an inverse in G , so there exists an operation $i \in \text{supp}(\nu)$ such that $i(D) = D'$ and such that the restriction of i to D' is the inverse of the diagonal of $h \circ g$. Hence $\{i \circ h \circ g\} \in \mathbb{G}^*$. Since $\text{supp}(\nu)$ forms a monoid under composition, $i \circ h \in \text{supp}(\nu)$ so we conclude that $\{i \circ h \circ g\} \in \text{supp}(\text{Exp}(\{g\}))$. \square

8. Characterisation of finite-valued languages. The goal of this section is to prove the characterisation of finite-valued languages solved by BLP. In particular, we prove the following theorem.

THEOREM 4 (restated). *Suppose that a finite-valued language Γ admits a symmetric fractional polymorphism of arity $m - 1 \geq 2$. Then Γ admits a symmetric fractional polymorphism of arity m .*

Let us fix a symmetric fractional polymorphism $\omega : \mathcal{O}^{(m-1)} \rightarrow \mathbb{R}_{\geq 0}$ of Γ of arity $m - 1$. We will use the letter s for operations in $\text{supp}(\omega)$ to emphasize that these operations are symmetric.

A symmetric fractional polymorphism of Γ of arity m will be constructed in two steps. The first one will rely on the Expansion Lemma. Essentially, in this step we start with a fractional polymorphism $\rho_0 = \frac{1}{m}(e_1^{(m)} + \dots + e_m^{(m)})$ and repeatedly modify it by applying the fractional polymorphism ω . The example below demonstrates such modification for $m = 3$.

EXAMPLE 7. *Suppose that language Γ admits a binary symmetric fractional polymorphism ω . For a function $f \in \Gamma$ and labellings $x, y, z \in \text{dom } f$ we can write*

$$\begin{aligned} f^3(x, y, z) &= \frac{1}{3}f^2(y, z) + \frac{1}{3}f^2(x, z) + \frac{1}{3}f^2(x, y) \\ &\geq \frac{1}{3} \sum_{s \in \text{supp}(\omega)} \omega(s)f(s(y, z)) + \frac{1}{3} \sum_{s \in \text{supp}(\omega)} \omega(s)f(s(x, z)) + \frac{1}{3} \sum_{s \in \text{supp}(\omega)} \omega(s)f(s(x, y)) \\ &= \sum_{s \in \text{supp}(\omega)} \omega(s)f^3(s(y, z), s(x, z), s(x, y)) \end{aligned}$$

This means that the following vector is a fractional polymorphism of Γ :

$$\rho_1 = \sum_{s \in \text{supp}(\omega)} \omega(s) \cdot \frac{1}{3}(\chi_{s \circ (e_2^{(3)}, e_3^{(3)})} + \chi_{s \circ (e_1^{(3)}, e_3^{(3)})} + \chi_{s \circ (e_1^{(3)}, e_2^{(3)})})$$

We can then take one component $\frac{1}{3}(\chi_{g_1} + \chi_{g_2} + \chi_{g_3})$ of the sum above and replace it with another vector by applying ω in a similar way. This shows how we can derive new fractional polymorphisms of Γ . Note that such polymorphisms will have a special structure, namely they will be a weighted sum of vectors of the form $\frac{1}{3}(\chi_{g_1} + \chi_{g_2} + \chi_{g_3})$ where $g_1, g_2, g_3 \in \mathcal{O}^{(3)}$. This means that we will be working with the set \mathbb{G} containing triplets of operations $\mathbf{g} = (g_1, g_2, g_3) \in \mathcal{O}^{(3 \rightarrow 3)}$. Recall that in Section 5 a probability distribution over such \mathbb{G} was called a fractional polymorphism of arity $3 \rightarrow 3$.

The example above can be generalised to other values of $m \geq 3$ in a natural way. The output of the first step (described in Section 8.1 below) will thus be a generalised fractional polymorphism of Γ of arity $m \rightarrow m$ with certain properties that will be exploited in step 2.

In the second step (Sections 8.2-8.5) we will turn it into an m -ary symmetric fractional polymorphism of Γ using tools such as Farkas' lemma. Note that in the second step the assumption that Γ is finite-valued will be essential.

With this introduction, we now proceed with the formal proof of Theorem 4.

8.1. Proof of Theorem 4: Step 1. We start with some additional notation and definitions. We use $[m]$ to denote set $\{1, \dots, m\}$. Let $\pi \in S_m$ be a permutation of $[m]$. For a labelling $\alpha = (a_1, \dots, a_m) \in D^m$ we define $\alpha^\pi \in D^m$ as follows: $\alpha^\pi = (a_{\pi(1)}, \dots, a_{\pi(m)})$. For an operation $g : D^m \rightarrow D$, let $g^\pi : D^m \rightarrow D$ be the following operation:

$$g^\pi(\alpha) = g(\alpha^\pi) \quad (8.1)$$

For a symmetric operation $s \in \text{supp}(\omega)$ of arity $m - 1$ we introduce the following definitions. For a labelling $\alpha = (a_1, \dots, a_m) \in D^m$ let $\alpha^s \in D^m$ be the labelling

$$\alpha^s = (s(\alpha_{-1}), \dots, s(\alpha_{-m})) \quad (8.2)$$

where $\alpha_{-i} \in D^{m-1}$ is the labelling obtained from α by removing the i -th element. For a mapping $\mathbf{g} : D^m \rightarrow D^m$, let $\mathbf{g}^s : D^m \rightarrow D^m$ be the mapping

$$\mathbf{g}^s(\alpha) = [\mathbf{g}(\alpha)]^s \quad (8.3a)$$

The last definition can also be expressed as

$$\mathbf{g}^s = (s \circ \mathbf{g}_{-1}, \dots, s \circ \mathbf{g}_{-m}) \quad (8.3b)$$

where $\mathbf{g}_{-i} : D^m \rightarrow D^{m-1}$ is the sequence of $m - 1$ operations obtained from $\mathbf{g} = (g_1, \dots, g_m)$ by removing the i -th operation. We use $\mathbf{g}^{s_1 \dots s_k}$ to denote $(\dots (\mathbf{g}^{s_1}) \dots)^{s_k}$.

Let $\mathbf{1}$ be the identity mapping $D^m \rightarrow D^m$, and let $\mathbb{G} = \{\mathbf{1}^{s_1 \dots s_k} \mid s_1, \dots, s_k \in \text{supp}(\omega), k \geq 0\} \subseteq \mathcal{O}^{(m \rightarrow m)}$ be the set of all mappings that can be obtained from $\mathbf{1}$ by applying operations from $\text{supp}(\omega)$.

Graph on mappings. Let us define a directed weighted graph (\mathbb{G}, E, w) with the set of edges $E = \{(\mathbf{g}, \mathbf{g}^s) \mid \mathbf{g} \in \mathbb{G}, s \in \text{supp}(\omega)\}$ and positive weights $w(\mathbf{g}, \mathbf{h}) = \sum_{s \in \text{supp}(\omega): \mathbf{h} = \mathbf{g}^s} \omega(s)$ for $(\mathbf{g}, \mathbf{h}) \in E$. Clearly, we have

$$\sum_{\mathbf{h}: (\mathbf{g}, \mathbf{h}) \in E} w(\mathbf{g}, \mathbf{h}) = 1 \quad \forall \mathbf{g} \in \mathbb{G} \quad (8.4)$$

The graph (\mathbb{G}, E) can be decomposed into strongly connected components, yielding a directed acyclic graph (DAG) on these components. We define $\text{Sinks}(\mathbb{G}, E)$ to be

the set of those strongly connected components $\mathbb{H} \subseteq \mathbb{G}$ of (\mathbb{G}, E) that are sinks of this DAG (i.e. have no outgoing edges). Any DAG has at least one sink, therefore $\text{Sinks}(\mathbb{G}, E)$ is non-empty. We denote $\mathbb{G}^* = \bigcup_{\mathbb{H} \in \text{Sinks}(\mathbb{G}, E)} \mathbb{H} \subseteq \mathbb{G}$.

By applying the Expansion Lemma to the sets of collections $(\mathbb{G}, \mathbb{G}^*)$ defined above we can obtain the following result.

LEMMA 11. *There exists a generalised fractional polymorphism ρ^* of Γ of arity $m \rightarrow m$ with $\text{supp}(\rho^*) \subseteq \mathbb{G}^*$.*

Proof. Clearly, Γ admits a least one generalised fractional polymorphism ρ with $\text{supp}(\rho) \subseteq \mathbb{G}$, namely $\rho = \chi_{\mathbb{1}}$. It thus suffices to prove the existence of an expansion operator Exp which is valid for Γ and non-vanishing with respect to $(\mathbb{G}, \mathbb{G}^*)$.

Given a mapping $\mathbf{g} \in \mathbb{G}$, we define the probability distribution $\rho = \text{Exp}(\mathbf{g})$ as follows:

$$\rho = \sum_{s \in \text{supp}(\omega)} \omega(s) \chi_{\mathbf{g}^s}$$

Let us check that it is indeed valid for Γ . Consider a function $f \in \Gamma$ of arity n and labellings $x^1, \dots, x^m \in D^n$. Denote $(y^1, \dots, y^m) = \mathbf{g}(x^1, \dots, x^m)$. Then,

$$\begin{aligned} \sum_{\mathbf{h} \in \text{supp}(\rho)} \rho(\mathbf{h}) f^m(\mathbf{h}(x^1, \dots, x^m)) &= \sum_{s \in \text{supp}(\omega)} \omega(s) f^m(\mathbf{g}^s(x^1, \dots, x^m)) \\ &= \sum_{s \in \text{supp}(\omega)} \omega(s) \frac{1}{m} \sum_{i \in [m]} f(s((y^1, \dots, y^m)_{-i})) \\ &= \frac{1}{m} \sum_{i \in [m]} \sum_{s \in \text{supp}(\omega)} \omega(s) f(s((y^1, \dots, y^m)_{-i})) \\ &\leq \frac{1}{m} \sum_{i \in [m]} f^{m-1}((y^1, \dots, y^m)_{-i}) \\ &= f^m(y^1, \dots, y^m) = f^m(\mathbf{g}(x^1, \dots, x^m)). \end{aligned}$$

Now let us show that the expansion operator Exp is non-vanishing. Observe that $\text{supp}(\text{Exp}(\mathbf{g})) = \{\mathbf{h} \mid (\mathbf{g}, \mathbf{h}) \in E\}$ for any $\mathbf{g} \in \mathbb{G}$. Furthermore, it follows from the definition of \mathbb{G}^* that for any $\mathbf{g} \in \mathbb{G}$ there exists a path in (\mathbb{G}, E) from \mathbf{g} to some node $\mathbf{g}^* \in \mathbb{G}^*$. These two facts imply the claim. \square

This concludes the first step of the proof. To summarize, we have constructed a generalised fractional polymorphism ρ^* of Γ with $\text{supp}(\rho^*) \subseteq \mathbb{G}^*$. Note that operations in collections $\mathbf{g} \in \text{supp}(\rho^*)$ are not necessarily symmetric (otherwise this would be a contradiction to Example 5). In the second step we will show that for finite-valued languages we can replace these collections with $\mathbf{p} \circ \mathbf{g}$, where $\mathbf{p} : D^m \rightarrow D^m$ is a mapping that orders tuples $\alpha = (a_1, \dots, a_m) \in D^m$ according to some total order on D . More precisely, we will show that

$$f^m(\mathbf{g}(x^1, \dots, x^m)) = f^m((\mathbf{p} \circ \mathbf{g})(x^1, \dots, x^m)) \quad \forall \mathbf{g} \in \mathbb{G}^*, f \in \Gamma, x^1, \dots, x^m \in \text{dom } f$$

This will imply that vector $\rho = \sum_{\mathbf{g} \in \text{supp}(\rho^*)} \rho^*(\mathbf{g}) \chi_{\mathbf{p} \circ \mathbf{g}}$ is also a generalised fractional polymorphism of Γ , which gives an m -ary symmetric fractional polymorphism of Γ , thus proving Theorem 4.

8.2. Proof of Theorem 4: Step 2. We start with the following observation.

PROPOSITION 12. Every $\mathbf{g} = (g_1, \dots, g_m) \in \mathbb{G}$ satisfies the following:

$$(g_1^\pi, \dots, g_m^\pi) = (g_{\pi(1)}, \dots, g_{\pi(m)}) \quad \forall \text{ permutations } \pi \in S_m \quad (8.5)$$

Thus, permuting the arguments of $g_i(\cdot, \dots, \cdot)$ gives a mapping which is also present in the sequence \mathbf{g} , possibly at a different position.

Proof. Checking that $\mathbf{1}$ satisfies (8.5) is straightforward. Let us prove that for any $\mathbf{g} : D^m \rightarrow D^m$ satisfying (8.5) and for any symmetric operation $s \in \mathcal{O}^{(m-1)}$, the mapping \mathbf{g}^s also satisfies (8.5). Consider $i \in [m]$. We need to show that $(s \circ \mathbf{g}_{-i})^\pi = s \circ \mathbf{g}_{-\pi(i)}$. For each $\alpha \in D^m$ we have

$$\begin{aligned} (s \circ \mathbf{g}_{-i})^\pi(\alpha) &= s \circ \mathbf{g}_{-i}(\alpha^\pi) \\ &= s(g_1(\alpha^\pi), \dots, g_{i-1}(\alpha^\pi), g_{i+1}(\alpha^\pi), \dots, g_m(\alpha^\pi)) \\ &= s(g_1^\pi(\alpha), \dots, g_{i-1}^\pi(\alpha), g_{i+1}^\pi(\alpha), \dots, g_m^\pi(\alpha)) \\ &= s(g_{\pi(1)}(\alpha), \dots, g_{\pi(i-1)}(\alpha), g_{\pi(i+1)}(\alpha), \dots, g_{\pi(m)}(\alpha)) = s \circ \mathbf{g}_{-\pi(i)}(\alpha) \end{aligned}$$

□

For the next statement consider a connected component $\mathbb{H} \in \text{Sinks}(\mathbb{G}, E)$, and denote $I = \mathbb{H} \times [m]$. Below we use the Iverson bracket notation: $[\phi] = 1$ if ϕ is true, and $[\phi] = 0$ otherwise.

LEMMA 13. (a) For any fixed distinct $\mathbf{g}', \mathbf{g}'' \in \mathbb{H}$, there exists a vector $\lambda \in \mathbb{R}_{\geq 0}^I$ that satisfies

$$\sum_{\mathbf{h}: (\mathbf{h}, \mathbf{g}) \in E} w(\mathbf{h}, \mathbf{g}) \lambda_{\mathbf{h}i} - \sum_{j \in [m] - \{i\}} \frac{\lambda_{\mathbf{g}j}}{m-1} = c_{\mathbf{g}} \quad \forall (\mathbf{g}, i) \in I \quad (8.6)$$

where $c_{\mathbf{g}} = [\mathbf{g} = \mathbf{g}'] - [\mathbf{g} = \mathbf{g}'']$.

(b) For any fixed distinct $i', i'' \in [m]$, there exists a vector $\lambda \in \mathbb{R}_{\geq 0}^{I \cup \mathbb{H}}$ that satisfies

$$\sum_{\mathbf{h}: (\mathbf{h}, \mathbf{g}) \in E} w(\mathbf{h}, \mathbf{g}) \lambda_{\mathbf{h}i} - \sum_{j \in [m] - \{i\}} \frac{\lambda_{\mathbf{g}j}}{m-1} = c_i \lambda_{\mathbf{g}} \quad \forall (\mathbf{g}, i) \in I \quad (8.7a)$$

$$\sum_{\mathbf{g} \in \mathbb{H}} \lambda_{\mathbf{g}} = 1 \quad (8.7b)$$

where $c_i = [i = i'] - [i = i'']$.

A proof of this statement is given in Section 8.3, and is based on Farkas' Lemma.

Now let us fix a function $f \in \Gamma$ of arity n . Given labellings x^1, \dots, x^m , we define labellings $x^{\mathbf{g}^i}$ for all $(\mathbf{g}, i) \in I$ via

$$(x^{\mathbf{g}^1}, \dots, x^{\mathbf{g}^m}) = \mathbf{g}(x^1, \dots, x^m) \quad (8.8)$$

Note that $x^{\mathbf{g}^i}$ is a function of (x^1, \dots, x^m) ; for brevity of notation, this dependence is not shown. For a vector $\lambda \in \mathbb{R}^{\mathbb{H}}$ and an index $i \in [m]$ we define the function F_i^λ via

$$F_i^\lambda(x^1, \dots, x^m) = \sum_{\mathbf{g} \in \mathbb{H}} \lambda_{\mathbf{g}} f(x^{\mathbf{g}^i}) \quad \forall x^1, \dots, x^m \in D^n \quad (8.9)$$

LEMMA 14. (a) It holds that $f^m(x^{\mathbf{g}'^1}, \dots, x^{\mathbf{g}'^m}) = f^m(x^{\mathbf{g}''^1}, \dots, x^{\mathbf{g}''^m})$ for all $\mathbf{g}', \mathbf{g}'' \in \mathbb{H}$ and $x^1, \dots, x^m \in D^n$.

(b) There exists a probability distribution λ over \mathbb{H} such that $F_i^\lambda(x^1, \dots, x^m) = F_{i'}^\lambda(x^1, \dots, x^m)$ for all $i', i'' \in [m]$ and $x^1, \dots, x^m \in D^n$.

A proof of Lemma 14 is given in section 8.4. The idea of the proof is as follows. Let us fix $\mathbf{g}', \mathbf{g}''$ and labellings $x^1, \dots, x^m \in D^n$. We will write down inequalities for the fractional polymorphism ω applied to $m-1$ labellings $(x^{\mathbf{g}^1}, \dots, x^{\mathbf{g}^m})_{-i}$ with $(\mathbf{g}, i) \in I$. We will then take a linear combination of these inequalities with weights $\lambda_{\mathbf{g}i} \geq 0$ constructed in Lemma 13(a); this will give inequality $f^m(x^{\mathbf{g}'^1}, \dots, x^{\mathbf{g}''^m}) \leq f^m(x^{\mathbf{g}''^1}, \dots, x^{\mathbf{g}''^m})$. This inequality should hold for all choices of $\mathbf{g}', \mathbf{g}''$, therefore it must actually be an equality. Part (b) of Lemma 14 will be proved in a similar way.

With Lemma 14 we will finally be able to prove the following (see Section 8.5).

LEMMA 15. Let \mathbf{g}^* be a mapping in \mathbb{G}^* and $\mathbf{p} \in \mathcal{O}^{(m \rightarrow m)}$ be any mapping such that $\mathbf{p}(\alpha)$ is a permutation of α for all $\alpha \in D^m$. Denote

$$\text{Range}_n(\mathbf{g}^*) = \{\mathbf{g}^*(x^1, \dots, x^m) \mid x^1, \dots, x^m \in D^n\}$$

For any function $f \in \Gamma$ of arity n and any $(x^1, \dots, x^m) \in \text{Range}_n(\mathbf{g}^*)$ it holds that $f^m(x^1, \dots, x^m) = f^m(\mathbf{p}(x^1, \dots, x^m))$.

This will imply Theorem 4. Indeed, we can construct an m -ary symmetric fractional polymorphism of Γ as follows. Take the vector ρ^* from Lemma 11, take a mapping $\mathbf{p} \in \mathcal{O}^{(m \rightarrow m)}$ that orders tuples $\alpha = (a_1, \dots, a_m) \in D^m$ according to some total order on D , and define the following vector.

$$\rho = \sum_{\mathbf{g} \in \text{supp}(\rho^*)} \rho^*(\mathbf{g}) \chi_{\mathbf{p} \circ \mathbf{g}}$$

Then, Γ admits ρ since for any $f \in \Gamma$ and for any labellings $x^1, \dots, x^m \in D^n$ we have

$$\begin{aligned} \sum_{\mathbf{h} \in \text{supp}(\rho)} \rho(\mathbf{h}) f^m(\mathbf{h}(x^1, \dots, x^m)) &= \sum_{\mathbf{g} \in \text{supp}(\rho^*)} \rho^*(\mathbf{g}) f^m(\mathbf{p}(\mathbf{g}(x^1, \dots, x^m))) \\ &= \sum_{\mathbf{g} \in \text{supp}(\rho^*)} \rho^*(\mathbf{g}) f^m(\mathbf{g}(x^1, \dots, x^m)) \leq f^m(x^1, \dots, x^m) \end{aligned}$$

Note, for any $\mathbf{h} = (h_1, \dots, h_m) \in \text{supp}(\rho)$, the operations h_1, \dots, h_m are symmetric. Indeed, we have $\mathbf{h} = \mathbf{p} \circ \mathbf{g}$ for some $\mathbf{g} \in \mathbb{G}^*$. If $\alpha \in D^m$ and π is a permutation of the set $[m]$ then $\mathbf{h}(\alpha^\pi) = \mathbf{p}(\mathbf{g}(\alpha^\pi)) = \mathbf{p}([\mathbf{g}(\alpha)]^\pi) = \mathbf{p}(\mathbf{g}(\alpha)) = \mathbf{h}(\alpha)$ which implies the claim.

A symmetric m -ary fractional polymorphism of Γ is finally given by $\omega_m = \sum_{\mathbf{g} \in \mathbb{G}} \rho(\mathbf{g}) \sum_{g \in \mathbf{g}} \frac{1}{m} \chi_g$.

It remains to prove Lemmas 13, 14 and 15.

8.3. Proof of Lemma 13. The lemma has two parts. For each part we will use a similar technique, namely we will assume the opposite and then derive a contradiction by using Farkas' lemma.

LEMMA 13 (restated). (a) There exists a vector $\lambda \in \mathbb{R}_{\geq 0}^I$ that satisfies

$$\sum_{\mathbf{h}: (\mathbf{h}, \mathbf{g}) \in E} w(\mathbf{h}, \mathbf{g}) \lambda_{\mathbf{h}i} - \sum_{j \in [m] - \{i\}} \frac{\lambda_{\mathbf{g}j}}{m-1} = c_{\mathbf{g}} \quad \forall (\mathbf{g}, i) \in I \quad (8.10)$$

where $c_{\mathbf{g}} = [\mathbf{g} = \mathbf{g}'] - [\mathbf{g} = \mathbf{g}'']$.

Proof. Suppose that the claim does not hold. By Farkas' lemma [64] there exists a vector $y \in \mathbb{R}^I$ such that

$$\sum_{(\mathbf{g}, i) \in I} c_{\mathbf{g}} y_{\mathbf{g}i} < 0 \quad (8.11a)$$

$$\sum_{\mathbf{h}: (\mathbf{g}, \mathbf{h}) \in E} w(\mathbf{g}, \mathbf{h}) y_{\mathbf{h}i} - \sum_{j \in [m] - \{i\}} \frac{y_{\mathbf{g}j}}{m-1} \geq 0 \quad \forall (\mathbf{g}, i) \in I \quad (8.11b)$$

Denote $u_{\mathbf{g}} = \sum_{i \in [m]} y_{\mathbf{g}i}$. Summing inequalities (8.11b) over $i \in [m]$ gives

$$\sum_{\mathbf{h}: (\mathbf{g}, \mathbf{h}) \in E} w(\mathbf{g}, \mathbf{h}) u_{\mathbf{h}} - u_{\mathbf{g}} \geq 0 \quad \forall \mathbf{g} \in \mathbb{H} \quad (8.12)$$

Denote $\mathbb{H}^* = \arg \max \{u_{\mathbf{g}} \mid \mathbf{g} \in \mathbb{H}\}$. From (8.4) and (8.12) we conclude that $\mathbf{g} \in \mathbb{H}^*$ implies $\mathbf{h} \in \mathbb{H}^*$ for all $(\mathbf{g}, \mathbf{h}) \in E$. Therefore, $\mathbb{H}^* = \mathbb{H}$ (since \mathbb{H} is a strongly connected component of G).

We showed that $u_{\mathbf{g}} = C$ for all $\mathbf{g} \in \mathbb{H}$ where $C \in \mathbb{R}$ is some constant. But then the expression on the LHS of (8.11a) equals $C - C = 0$, a contradiction. \square

LEMMA 13 (restated). (b) *There exists a vector $\lambda \in \mathbb{R}_{\geq 0}^{I \cup \mathbb{H}}$ that satisfies*

$$\sum_{\mathbf{h}: (\mathbf{h}, \mathbf{g}) \in E} w(\mathbf{h}, \mathbf{g}) \lambda_{\mathbf{h}i} - \sum_{j \in [m] - \{i\}} \frac{\lambda_{\mathbf{g}j}}{m-1} = c_i \lambda_{\mathbf{g}} \quad \forall (\mathbf{g}, i) \in I \quad (8.13a)$$

$$\sum_{\mathbf{g} \in \mathbb{H}} \lambda_{\mathbf{g}} = 1 \quad (8.13b)$$

where $c_i = [i = i'] - [i = i'']$.

Proof. Suppose that the claim does not hold. By Farkas' lemma [64] there exist a vector $y \in \mathbb{R}^I$ and a scalar $z \in \mathbb{R}$ such that

$$z < 0 \quad (8.14a)$$

$$z - \sum_{i \in [m]} c_i y_{\mathbf{g}i} \geq 0 \quad \forall \mathbf{g} \in \mathbb{H} \quad (8.14b)$$

$$\sum_{\mathbf{h}: (\mathbf{g}, \mathbf{h}) \in E} w(\mathbf{g}, \mathbf{h}) y_{\mathbf{h}i} - \sum_{j \in [m] - \{i\}} \frac{y_{\mathbf{g}j}}{m-1} \geq 0 \quad \forall (\mathbf{g}, i) \in I \quad (8.14c)$$

Denote $u_{\mathbf{g}} = \sum_{i \in [m]} y_{\mathbf{g}i}$. Using the same argument as in part (a) we conclude that $u_{\mathbf{g}} = C$ for all $\mathbf{g} \in \mathbb{H}$ where $C \in \mathbb{R}$ is some constant. We can assume w.l.o.g. that this constant is zero. Indeed, this can be achieved by subtracting C/m from values $y_{\mathbf{g}i}$ for all $(\mathbf{g}, i) \in I$ with $\mathbf{g} \in \mathbb{H}$; it can be checked (using eq. (8.4)) that this operation preserves inequalities (8.14). We thus have

$$\sum_{j \in [m] - \{i\}} y_{\mathbf{g}j} = -y_{\mathbf{g}i} \quad \forall (\mathbf{g}, i) \in I \quad (8.15)$$

Substituting this into (8.14c) gives

$$\sum_{\mathbf{h}: (\mathbf{g}, \mathbf{h}) \in E} w(\mathbf{g}, \mathbf{h}) y_{\mathbf{h}i} + \frac{y_{\mathbf{g}i}}{m-1} \geq 0 \quad \forall (\mathbf{g}, i) \in I \quad (8.16)$$

Consider $k \in [m]$. Summing (8.16) over $i \in [m] - \{k\}$ and then using (8.15) yields

$$\sum_{\mathbf{h}: (\mathbf{g}, \mathbf{h}) \in E} w(\mathbf{g}, \mathbf{h})(-y_{\mathbf{h}k}) + \frac{-y_{\mathbf{g}k}}{m-1} \geq 0 \quad \forall (\mathbf{g}, k) \in I \quad (8.17)$$

Combining (8.16) and (8.17) gives

$$\sum_{\mathbf{h}: (\mathbf{g}, \mathbf{h}) \in E} w(\mathbf{g}, \mathbf{h})y_{\mathbf{h}i} + \frac{y_{\mathbf{g}i}}{m-1} = 0 \quad \forall (\mathbf{g}, i) \in I \quad (8.18)$$

Denote $r_{\mathbf{g}} = \sum_{i \in [m]} c_i y_{\mathbf{g}i}$ for $\mathbf{g} \in \mathbb{H}$. Summing (8.18) over $i \in [m]$ with appropriate coefficients gives

$$\sum_{\mathbf{h}: (\mathbf{g}, \mathbf{h}) \in E} w(\mathbf{g}, \mathbf{h})r_{\mathbf{h}} + \frac{r_{\mathbf{g}}}{m-1} = 0 \quad \forall \mathbf{g} \in \mathbb{H} \quad (8.19)$$

From (8.14a) and (8.14b) we conclude that $r_{\mathbf{g}} < 0$ for all $\mathbf{g} \in \mathbb{H}$, and thus eq. (8.19) cannot hold, a contradiction. \square

8.4. Proof of Lemma 14. Let us fix a function $f \in \Gamma$ of arity n and a connected component $\mathbb{H} \in \text{Sinks}(\mathbb{G}, E)$. In this subsection we will prove the following.

LEMMA 14 (equivalent statement). (a) *Inequality*

$$\sum_{i \in [m]} f(x^{\mathbf{g}'i}) - \sum_{i \in [m]} f(x^{\mathbf{g}''i}) \leq 0 \quad (8.20a)$$

holds for any distinct mappings $\mathbf{g}', \mathbf{g}'' \in \mathbb{H}$ and any $x^1, \dots, x^m \in D^n$.

(b) *There exists a probability distribution λ over \mathbb{H} such that*

$$\sum_{\mathbf{g} \in \mathbb{H}} \lambda_{\mathbf{g}} f(x^{\mathbf{g}'i}) - \sum_{\mathbf{g} \in \mathbb{H}} \lambda_{\mathbf{g}} f(x^{\mathbf{g}''i}) \leq 0 \quad (8.20b)$$

for any distinct indices $i', i'' \in [m]$ and any $x^1, \dots, x^m \in D^n$.

Since inequality (8.20a) holds for any pair of distinct mappings $\mathbf{g}', \mathbf{g}'' \in \mathbb{H}$, we conclude that in (8.20a) we actually must have an equality (and similarly for (8.20b)). Therefore, the statement above is indeed equivalent to the original formulation of Lemma 14, which had equalities. (Note, we have also moved terms from the right-hand side of the original equalities to the left-hand side with the negative sign; for that we have used the fact the Γ is finite-valued.)

We will need the following observation.

PROPOSITION 16. *If $\mathbf{h} = \mathbf{g}^s$ where $\mathbf{g} \in \mathbb{H}$, $s \in \text{supp}(\omega)$ then $x^{\mathbf{h}i} = s((x^{\mathbf{g}^1}, \dots, x^{\mathbf{g}^m})_{-i})$ for $i \in [m]$ where $(x^{\mathbf{g}^1}, \dots, x^{\mathbf{g}^m})_{-i}$ is the sequence of $m-1$ labellings obtained by removing the i -th labelling.*

Proof. Consider a coordinate $v \in [n]$, and denote $\alpha = (x_v^1, \dots, x_v^m)$, $\beta = (x_v^{\mathbf{g}^1}, \dots, x_v^{\mathbf{g}^m})$, $\gamma = (x_v^{\mathbf{h}^1}, \dots, x_v^{\mathbf{h}^m})$. By definition (8.8), $\beta = \mathbf{g}(\alpha)$ and $\gamma = \mathbf{h}(\alpha)$. Therefore, $\gamma = \mathbf{g}^s(\alpha) = [\mathbf{g}(\alpha)]^s = \beta^s$. In other words, the i th component of γ equals $s(\beta_{-i})$, which is what we needed to show. \square

Consider $m-1$ labellings $(x^{\mathbf{g}^1}, \dots, x^{\mathbf{g}^m})_{-i}$ for $(\mathbf{g}, i) \in I$. Applying the polymorphism inequality to these labellings gives

$$\sum_{s \in \text{supp}(\omega)} \omega(s) f(s((x^{\mathbf{g}^1}, \dots, x^{\mathbf{g}^m})_{-i})) \leq \frac{1}{m-1} \sum_{j \in [m] - \{i\}} f(x^{\mathbf{g}^j})$$

Let us multiply this inequality by weight $\lambda_{\mathbf{g}i} \geq 0$ (to be defined later), and apply Proposition 16 and the fact that $w(\mathbf{g}, \mathbf{h}) = \sum_{s \in \text{supp}(\omega): \mathbf{h}=\mathbf{g}^s} \omega(s)$:

$$\lambda_{\mathbf{g}i} \sum_{\mathbf{h}: (\mathbf{g}, \mathbf{h}) \in E} w(\mathbf{g}, \mathbf{h}) f(x^{\mathbf{h}i}) - \frac{\lambda_{\mathbf{g}i}}{m-1} \sum_{j \in [m] - \{i\}} f(x^{\mathbf{g}j}) \leq 0 \quad \forall (\mathbf{g}, i) \in I$$

Summing these inequalities over $(\mathbf{g}, i) \in I$ gives

$$\sum_{(\mathbf{g}, i) \in I} \left[\sum_{\mathbf{h}: (\mathbf{h}, \mathbf{g}) \in E} w(\mathbf{h}, \mathbf{g}) \lambda_{\mathbf{h}i} - \sum_{j \in [m] - \{i\}} \frac{\lambda_{\mathbf{g}j}}{m-1} \right] f(x^{\mathbf{g}i}) \leq 0 \quad (8.21)$$

Plugging weights $\lambda_{\mathbf{g}i}$ from Lemma 13(a) into 8.21 gives inequality (8.20a). This proves Lemma 14(a).

Similarly, we can plug weights $\lambda_{\mathbf{g}i}$ from Lemma 13(b) into (8.21) and get inequality (8.20b). However, we need an additional argument in order to establish Lemma 14(b) with this strategy. Indeed, the vector λ in Lemma 13(b) depends on the pair (i', i'') ; let us denote it as $\lambda^{i' i''}$. We need to show that these vectors can be chosen in such a way that for a given $\mathbf{g} \in \mathbb{H}$, the components $\lambda_{\mathbf{g}}^{i' i''}$ are the same for all pairs (i', i'') . This can be done as follows. Take the vector λ^{12} constructed in Lemma 13(b). For a pair of distinct indices $(i', i'') \neq (1, 2)$ select a permutation π of $[m]$ with $\pi(i') = 1$, $\pi(i'') = 2$, and define vector $\lambda^{i' i''}$ via

$$\lambda_{\mathbf{g}}^{i' i''} = \lambda_{\mathbf{g}}^{12} \quad \forall \mathbf{g} \in \mathbb{H} \quad \lambda_{\mathbf{g}}^{i' i''} = \lambda_{\mathbf{g}^{\pi(i)}}^{12} \quad \forall (\mathbf{g}, i) \in I$$

Clearly, the vector $\lambda^{i' i''}$ satisfies conditions of Lemma 13(b) for the pair (i', i'') . Thus, Lemma 13(b) indeed implies Lemma 14(b).

8.5. Proof of Lemma 15. In this subsection we prove the following.

LEMMA 15 (restated). *Let \mathbf{g}^* be a mapping in \mathbb{G}^* and $\mathbf{p} \in \mathcal{O}^{(m \rightarrow m)}$ be any mapping such that $\mathbf{p}(\alpha)$ is a permutation of α for all $\alpha \in D^m$. Denote*

$$\text{Range}_n(\mathbf{g}^*) = \{\mathbf{g}^*(x^1, \dots, x^m) \mid x^1, \dots, x^m \in D^n\}$$

For any function $f \in \Gamma$ of arity n and any $(x^1, \dots, x^m) \in \text{Range}_n(\mathbf{g}^)$ it holds that $f^m(x^1, \dots, x^m) = f^m(\mathbf{p}(x^1, \dots, x^m))$.*

Fix function $f \in \Gamma$ of arity n , and let $\mathbb{H} \in \text{Sinks}(\mathbb{G}, E)$ be the strongly connected component that contains \mathbf{g}^* . Let $\lambda \in \mathbb{R}_{\geq 0}^{\mathbb{H}}$ be a vector constructed in Lemma 14(b). We denote $F^\lambda(x^1, \dots, x^m) = F_i^\lambda(x^1, \dots, x^m)$ for $i \in [m]$.

LEMMA 17. *The following transformation does not change $F^\lambda(x^1, \dots, x^m)$: pick a coordinate $v \in [n]$ and permute the labels (x_v^1, \dots, x_v^m) .*

Proof. It suffices to prove the claim for a permutation π which swaps the labels x_v^i and x_v^j for $i, j \in [m]$ (since any other permutation can be obtained by repeatedly applying such swaps). Since $m \geq 3$ there exists an index $k \in [m] - \{i, j\}$. We claim that for any $\mathbf{g} = (g_1, \dots, g_m) \in \mathbb{H}$, the labelling $x^{\mathbf{g}k} = g_k(x^1, \dots, x^m)$ is not affected by the swap above. Indeed, it suffices to check this for coordinate v (for other coordinates the claim is trivial). Denoting the new labellings as \tilde{x}^i and $\tilde{x}^{\mathbf{g}k}$, we can write

$$\begin{aligned} \tilde{x}_v^{\mathbf{g}k} &= g_k(\tilde{x}_v^1, \dots, \tilde{x}_v^m) = g_{\pi(k)}(\tilde{x}_v^1, \dots, \tilde{x}_v^m) && // \text{ since } \pi(k) = k \\ &= g_k^\pi(\tilde{x}_v^1, \dots, \tilde{x}_v^m) && // \text{ by Proposition 12} \\ &= g_k(\tilde{x}_v^{\pi(1)}, \dots, \tilde{x}_v^{\pi(m)}) = g_k(x_v^1, \dots, x_v^m) = x_v^{\mathbf{g}k} \end{aligned}$$

Since the labellings $x^{\mathbf{g}^k}$ do not change, the value of $F_k^\lambda(x^1, \dots, x^m)$ is also not affected by the swap (see its definition in eq. (8.9).) The lemma is proved. \square

LEMMA 18. *If $(x^1, \dots, x^m) \in \text{Range}_n(\mathbf{g}^*)$ then $(x^1, \dots, x^m) = (x^{\mathbf{g}^1}, \dots, x^{\mathbf{g}^m})$ for some $\mathbf{g} \in \mathbb{H}$.*

Proof. It suffices to show that there exists $\mathbf{g} \in \mathbb{H}$ with $\mathbf{g} \circ \mathbf{g}^* = \mathbf{g}^*$.

Note that $\mathbb{1}^{s_1 \dots s_k} \circ \mathbf{h} = \mathbf{h}^{s_1 \dots s_k}$ for any $s_1, \dots, s_k \in \text{supp}(\omega)$ and $\mathbf{h} \in \mathcal{O}^{(m \rightarrow m)}$, since for any $\alpha \in D^m$ we have

$$[\mathbb{1}^{s_1 \dots s_k} \circ \mathbf{h}](\alpha) = \mathbb{1}^{s_1 \dots s_k}(\mathbf{h}(\alpha)) = [\mathbb{1}(\mathbf{h}(\alpha))]^{s_1 \dots s_k} = [\mathbf{h}(\alpha)]^{s_1 \dots s_k} = \mathbf{h}^{s_1 \dots s_k}(\alpha).$$

Therefore, conditions $\mathbf{g} \in \mathbb{G}$, $\mathbf{h} \in \mathbb{H}$ imply that $\mathbf{g} \circ \mathbf{h} \in \mathbb{H}$ (since \mathbf{g} can be written as $\mathbf{g} = \mathbb{1}^{s_1 \dots s_k}$ and there are no edges leaving \mathbb{H}).

Since \mathbb{H} is strongly connected, there is a path in (\mathbb{G}, E) from $\mathbf{g}^* \circ \mathbf{g}^* \in \mathbb{H}$ to $\mathbf{g}^* \in \mathbb{H}$, i.e. $[\mathbf{g}^* \circ \mathbf{g}^*]^{s_1 \dots s_k} = \mathbf{g}^*$ for some $s_1, \dots, s_k \in \text{supp}(\omega)$. Equivalently, $\mathbf{h} \circ \mathbf{g}^* \circ \mathbf{g}^* = \mathbf{g}^*$ where $\mathbf{h} = \mathbb{1}^{s_1 \dots s_k}$. It can be checked that mapping $\mathbf{g} = \mathbf{h} \circ \mathbf{g}^*$ has the desired properties. \square

LEMMA 19. *If $(x^1, \dots, x^m) \in \text{Range}_n(\mathbf{g}^*)$ then $f^m(x^1, \dots, x^m) = F^\lambda(x^1, \dots, x^m)$.*

Proof. From Theorem 14(a) and Lemma 18 we get that $f^m(x^{\mathbf{g}^1}, \dots, x^{\mathbf{g}^m}) = f^m(x^1, \dots, x^m)$ for all $\mathbf{g} \in \mathbb{H}$. Using this fact and the definition of $F_i^\lambda(\cdot)$, we can write

$$\begin{aligned} F^\lambda(x^1, \dots, x^m) &= \frac{1}{m} \sum_{i \in [m]} F_i^\lambda(x^1, \dots, x^m) = \frac{1}{m} \sum_{\mathbf{g} \in \mathbb{H}} \lambda_{\mathbf{g}} \sum_{i \in [m]} f(x^{\mathbf{g}^i}) \\ &= \sum_{\mathbf{g} \in \mathbb{H}} \lambda_{\mathbf{g}} f^m(x^{\mathbf{g}^1}, \dots, x^{\mathbf{g}^m}) = \sum_{\mathbf{g} \in \mathbb{H}} \lambda_{\mathbf{g}} f^m(x^1, \dots, x^m) = f^m(x^1, \dots, x^m) \end{aligned}$$

The lemma follows. \square

We can finally establish Lemma 15. For labelings $(x^1, \dots, x^m) \in \text{Range}_n(\mathbf{g}^*)$ we can write

$$f^m(x^1, \dots, x^m) \stackrel{(1)}{=} F^\lambda(x^1, \dots, x^m) \stackrel{(2)}{=} F^\lambda(\mathbf{p}(x^1, \dots, x^m)) \stackrel{(3)}{=} f^m(\mathbf{p}(x^1, \dots, x^m))$$

where equalities (1) and (3) follow from Lemma 19, and (2) follows from Lemma 17. Note, to be able to apply Lemma 19 in (3), we need the condition $\mathbf{p}(x^1, \dots, x^m) \in \text{Range}_n(\mathbf{g}^*)$. The proof of this condition follows mechanically from the assumption $(x^1, \dots, x^m) \in \text{Range}_n(\mathbf{g}^*)$ and Proposition 12, and is omitted.

Acknowledgements. We thank Andrei Krokhin for helpful discussions and for communicating the result of Raghavendra [62]. We also thank the anonymous referees for their diligent work on improving the presentation of the paper.

REFERENCES

- [1] LIBOR BARTO, *The dichotomy for conservative constraint satisfaction problems revisited*, in Proceedings of the 26th IEEE Symposium on Logic in Computer Science (LICS'11), IEEE Computer Society, 2011, pp. 301–310.
- [2] LIBOR BARTO AND MARCIN KOZIK, *Robust Satisfiability of Constraint Satisfaction Problems*, in Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC'12), ACM, 2012, pp. 931–940.
- [3] ———, *Constraint Satisfaction Problems Solvable by Local Consistency Methods*, Journal of the ACM, 61 (2014). Article No. 3.
- [4] LIBOR BARTO, MARCIN KOZIK, AND TODD 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 (2009), pp. 1782–1802.

- [5] ANDREI BULATOV, *A dichotomy theorem for constraint satisfaction problems on a 3-element set*, Journal of the ACM, 53 (2006), pp. 66–120.
- [6] ANDREI BULATOV, ANDREI KROKHIN, AND PETER JEAVONS, *Classifying the Complexity of Constraints using Finite Algebras*, SIAM Journal on Computing, 34 (2005), pp. 720–742.
- [7] ANDREI A. BULATOV, *Complexity of conservative constraint satisfaction problems*, ACM Transactions on Computational Logic, 12 (2011). Article 24.
- [8] CHANDRA CHEKURI, SANJEEV KHANNA, JOSEPH NAOR, AND LEONID ZOSIN, *A linear programming formulation and approximation algorithms for the metric labeling problem*, SIAM Journal on Discrete Mathematics, 18 (2004), pp. 608–625.
- [9] DAVID A. COHEN, MARTIN C. COOPER, PÁIDÍ CREED, PETER JEAVONS, AND STANISLAV ŽIVNÝ, *An algebraic theory of complexity for discrete optimisation*, SIAM Journal on Computing, 42 (2013), pp. 915–1939.
- [10] DAVID A. COHEN, MARTIN C. COOPER, AND PETER G. JEAVONS, *An Algebraic Characterisation of Complexity for Valued Constraints*, in Proceedings of the 12th International Conference on Principles and Practice of Constraint Programming (CP’06), vol. 4204 of Lecture Notes in Computer Science, Springer, 2006, pp. 107–121.
- [11] ———, *Generalising submodularity and Horn clauses: Tractable optimization problems defined by tournament pair multimorphisms*, Theoretical Computer Science, 401 (2008), pp. 36–51.
- [12] DAVID A. COHEN, MARTIN C. COOPER, PETER G. JEAVONS, AND ANDREI A. KROKHIN, *The Complexity of Soft Constraint Satisfaction*, Artificial Intelligence, 170 (2006), pp. 983–1016.
- [13] DAVID A. COHEN, PÁIDÍ CREED, PETER G. JEAVONS, AND STANISLAV ŽIVNÝ, *An algebraic theory of complexity for valued constraints: Establishing a Galois connection*, in Proceedings of the 36th International Symposium on Mathematical Foundations of Computer Science (MFCS’11), vol. 6907 of Lecture Notes in Computer Science, Springer, 2011, pp. 231–242.
- [14] MARTIN C. COOPER, *Minimization of Locally Defined Submodular Functions by Optimal Soft Arc Consistency*, Constraints, 13 (2008), pp. 437–458.
- [15] MARTIN C. COOPER, SIMON DE GIVRY, MARTÍ SÁNCHEZ, THOMAS SCHIEX, MATTHIAS ZYTNICKI, AND TOMÁŠ WERNER, *Soft arc consistency revisited*, Artificial Intelligence, 174 (2010), pp. 449–478.
- [16] MARTIN C. COOPER AND THOMAS SCHIEX, *Arc consistency for soft constraints*, Artificial Intelligence, 154 (2004), pp. 199–227.
- [17] YVES CRAMA AND PETER L. HAMMER, *Boolean Functions - Theory, Algorithms, and Applications*, Cambridge University Press, 2011.
- [18] PÁIDÍ CREED AND STANISLAV ŽIVNÝ, *On minimal weighted clones*, in Proceedings of the 17th International Conference on Principles and Practice of Constraint Programming (CP’11), vol. 6876 of Lecture Notes in Computer Science, Springer, 2011, pp. 210–224.
- [19] NADIA CREIGNOU, SANJEEV KHANNA, AND MADHU SUDAN, *Complexity Classification of Boolean Constraint Satisfaction Problems*, vol. 7 of SIAM Monographs on Discrete Mathematics and Applications, SIAM, 2001.
- [20] VÍCTOR DALMAU, ANDREI KROKHIN, AND RAJSEKAR MANOKARAN, *Towards a characterization of constant-factor approximable Min CSPs*, in Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA’15), SIAM, 2015.
- [21] VÍCTOR DALMAU AND ANDREI A. KROKHIN, *Robust Satisfiability for CSPs: Hardness and Algorithmic Results*, ACM Transactions on Computation Theory, 5 (2013). Article No. 15.
- [22] VÍCTOR DALMAU AND JUSTIN PEARSON, *Set Functions and Width 1 Problems*, in Proceedings of the 5th International Conference on Constraint Programming (CP’99), vol. 1713 of Lecture Notes in Computer Science, Springer, 1999, pp. 159–173.
- [23] VLADIMIR DEINEKO, PETER JONSSON, MIKAEL KLASSON, AND ANDREI KROKHIN, *The approximability of Max CSP with fixed-value constraints*, Journal of the ACM, 55 (2008). Article 16.
- [24] ALINA ENE, JAN VONDRÁK, AND YI WU, *Local distribution and the symmetry gap: Approximability of multiway partitioning problems*, in Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA’13), SIAM, 2013, pp. 306–325.
- [25] TOMÁS FEDER AND MOSHE Y. VARDI, *The Computational Structure of Monotone Monadic SNP and Constraint Satisfaction: A Study through Datalog and Group Theory*, SIAM Journal on Computing, 28 (1998), pp. 57–104.
- [26] EUGENE C. FREUDER, *Synthesizing Constraint Expressions*, Communications of the ACM, 21 (1978), pp. 958–966.
- [27] SATORU FUJISHIGE AND SATORU IWATA, *Bisubmodular Function Minimization*, SIAM Journal on Discrete Mathematics, 19 (2005), pp. 1065–1073.
- [28] GEORG GOTTLÖB, GIANLUIGI GRECO, AND FRANCESCO SCARCELLO, *Tractable Optimization Problems through Hypergraph-Based Structural Restrictions*, in Proceedings of the 36th

- International Colloquium on Automata, Languages and Programming (ICALP'09), Part II, vol. 5556 of Lecture Notes in Computer Science, Springer, 2009, pp. 16–30.
- [29] IGOR GRIDCHYN AND VLADIMIR KOLMOGOROV, *Potts model, parametric maxflow and k -submodular functions*, in Proceedings of the 14th IEEE International Conference on Computer Vision (ICCV'13), IEEE, 2013, pp. 2320–2327.
- [30] MARTIN GROHE, *The complexity of homomorphism and constraint satisfaction problems seen from the other side*, Journal of the ACM, 54 (2007), pp. 1–24.
- [31] PAVOL HELL AND JAROSLAV NEŠETŘIL, *Colouring, constraint satisfaction, and complexity*, Computer Science Review, 2 (2008), pp. 143–163.
- [32] HIROSHI HIRAI, *Discrete Convexity and Polynomial Solvability in Minimum 0-Extension Problems*, in Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'13), SIAM, 2013, pp. 1770–1778.
- [33] ANNA HUBER AND VLADIMIR KOLMOGOROV, *Towards Minimizing k -Submodular Functions*, in Proceedings of the 2nd International Symposium on Combinatorial Optimization (ISCO'12), vol. 7422 of Lecture Notes in Computer Science, Springer, 2012, pp. 451–462.
- [34] ANNA HUBER, ANDREI KROKHIN, AND ROBERT POWELL, *Skew bisubmodularity and valued CSPs*, SIAM Journal on Computing, 43 (2014), pp. 1064–1084.
- [35] PAWEŁ M. IDZIAK, PETAR MARKOVIC, RALPH MCKENZIE, MATTHEW VALERIOTE, AND ROSS WILLARD, *Tractability and learnability arising from algebras with few subpowers*, SIAM Journal on Computing, 39 (2010), pp. 3023–3037.
- [36] SATORU IWATA, *Submodular Function Minimization*, Mathematical Programming, 112 (2008), pp. 45–64.
- [37] SATORU IWATA, LISA FLEISCHER, AND SATORU FUJISHIGE, *A combinatorial strongly polynomial algorithm for minimizing submodular functions*, Journal of the ACM, 48 (2001), pp. 761–777.
- [38] PETER JEAVONS, ANDREI KROKHIN, AND STANISLAV ŽIVNÝ, *The complexity of valued constraint satisfaction*, Bulletin of the European Association for Theoretical Computer Science (EATCS), 113 (2014), pp. 21–55.
- [39] PETER G. JEAVONS, *On the Algebraic Structure of Combinatorial Problems*, Theoretical Computer Science, 200 (1998), pp. 185–204.
- [40] PETER G. JEAVONS, DAVID A. COHEN, AND MARC GYSSENS, *Closure Properties of Constraints*, Journal of the ACM, 44 (1997), pp. 527–548.
- [41] JASON K. JOHNSON, DMITRY M. MALIOUTOV, AND ALAN S. WILLSKY, *Lagrangian relaxation for MAP estimation in graphical models*, in Allerton Conference on Communication, Control and Computing, 2007, pp. 64–73.
- [42] PETER JONSSON, ANDREI A. KROKHIN, AND FREDRIK KUIVINEN, *Hard constraint satisfaction problems have hard gaps at location 1*, Theoretical Computer Science, 410 (2009), pp. 3856–3874.
- [43] PETER JONSSON, FREDRIK KUIVINEN, AND JOHAN THAPPER, *Min CSP on Four Elements: Moving Beyond Submodularity*, in Proceedings of the 17th International Conference on Principles and Practice of Constraint Programming (CP'11), vol. 6876 of Lecture Notes in Computer Science, Springer, 2011, pp. 438–453.
- [44] SANJEEV KHANNA, MADHU SUDAN, LUCA TREVISAN, AND DAVID WILLIAMSON, *The approximability of constraint satisfaction problems*, SIAM Journal on Computing, 30 (2001), pp. 1863–1920.
- [45] SUBHASH KHOT, *On the unique games conjecture (invited survey)*, in Proceedings of the 25th Annual IEEE Conference on Computational Complexity (CCC'10), IEEE Computer Society, 2010, pp. 99–121.
- [46] CARLETON L. KINGSFORD, BERNARD CHAZELLE, AND MONA SINGH, *Solving and analyzing side-chain positioning problems using linear and integer programming*, Bioinformatics, 21 (2005), pp. 1028–1039.
- [47] PHOKION G. KOLAITIS AND MOSHE Y. VARDI, *Conjunctive-Query Containment and Constraint Satisfaction*, Journal of Computer and System Sciences, 61 (2000), pp. 302–332.
- [48] VLADIMIR KOLMOGOROV, *Submodularity on a tree: Unifying l^k -convex and bisubmodular functions*, in Proceedings of the 36th International Symposium on Mathematical Foundations of Computer Science (MFCS'11), vol. 6907 of Lecture Notes in Computer Science, Springer, 2011, pp. 400–411.
- [49] ———, *The power of linear programming for finite-valued CSPs: a constructive characterization*, in Proceedings of the 40th International Colloquium on Automata, Languages and Programming (ICALP'13), vol. 7965 of Lecture Notes in Computer Science, Springer, 2013, pp. 625–636.
- [50] VLADIMIR KOLMOGOROV AND STANISLAV ŽIVNÝ, *The complexity of conservative valued CSPs*, Journal of the ACM, 60 (2013). Article 10.

- [51] N. KOMODAKIS, N. PARAGIOS, AND G. TZIRITAS, *MRF energy minimization and beyond via dual decomposition*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 33 (2011), pp. 531–552.
- [52] ARIE KOSTER, STAN P.M. VAN HOESSEL, AND ANTOON W.J. KOLEN, *The partial constraint satisfaction problem: Facets and lifting theorems*, Operations Research Letters, 23 (1998), pp. 89–97.
- [53] ANDREI KROKHIN AND BENOIT LAROSE, *Maximizing Supermodular Functions on Product Lattices, with Application to Maximum Constraint Satisfaction*, SIAM Journal on Discrete Mathematics, 22 (2008), pp. 312–328.
- [54] FREDRIK KUIVINEN, *On the complexity of submodular function minimisation on diamonds*, Discrete Optimization, 8 (2011), pp. 459–477.
- [55] GÁBOR KUN, RYAN O'DONNELL, SUGURU TAMAKI, YUICHI YOSHIDA, AND YUAN ZHOU, *Linear programming, width-1 CSPs, and robust satisfaction*, in Proceedings of the 3rd Innovations in Theoretical Computer Science (ITCS'12), ACM, 2012, pp. 484–495.
- [56] STEFFEN L. LAURITZEN, *Graphical Models*, Oxford University Press, 1996.
- [57] ALAN K. MACKWORTH, *Consistency in Networks of Relations*, Artificial Intelligence, 8 (1977), pp. 99–118.
- [58] DÁNIEL MARX, *Tractable hypergraph properties for constraint satisfaction and conjunctive queries*, Journal of the ACM, 60 (2013). Article No. 42.
- [59] S. THOMAS MCCORMICK AND SATORU FUJISHIGE, *Strongly polynomial and fully combinatorial algorithms for bisubmodular function minimization*, Mathematical Programming, 122 (2010), pp. 87–120.
- [60] UGO MONTANARI, *Networks of Constraints: Fundamental properties and applications to picture processing*, Information Sciences, 7 (1974), pp. 95–132.
- [61] PRASAD RAGHAVENDRA, *Optimal algorithms and inapproximability results for every CSP?*, in Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC'08), ACM, 2008, pp. 245–254.
- [62] ———, *Approximating NP-hard Problems: Efficient Algorithms and their Limits*, PhD Thesis, (2009).
- [63] THOMAS J. SCHAEFER, *The Complexity of Satisfiability Problems*, in Proceedings of the 10th Annual ACM Symposium on Theory of Computing (STOC'78), ACM, 1978, pp. 216–226.
- [64] ALEXANDER SCHRIJVER, *Theory of linear and integer programming*, John Wiley & Sons, Inc., 1986.
- [65] ———, *A Combinatorial Algorithm Minimizing Submodular Functions in Strongly Polynomial Time*, Journal of Combinatorial Theory, Series B, 80 (2000), pp. 346–355.
- [66] H. D. SHERALI AND W. P. ADAMS, *A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems*, SIAM Journal of Discrete Mathematics, 3 (1990), pp. 411–430.
- [67] MICHAEL I. SHLEZINGER, *Syntactic analysis of two-dimensional visual signals in noisy conditions*, Cybernetics and Systems Analysis, 12 (1976), pp. 612–628. Translation from Russian.
- [68] DAVID SONTAG, AMIR GLOBERSON, AND TOMMI JAAKKOLA, *Introduction to dual decomposition for inference*, in Optimization for Machine Learning, Suvrit Sra, Sebastian Nowozin, and Stephen J. Wright, eds., MIT Press, 2011.
- [69] JOHAN THAPPER AND STANISLAV ŽIVNÝ, *The power of linear programming for valued CSPs*, in Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS'12), IEEE, 2012, pp. 669–678.
- [70] ———, *The complexity of finite-valued CSPs*, in Proceedings of the 45th ACM Symposium on the Theory of Computing (STOC'13), ACM, 2013, pp. 695–704.
- [71] HANNES UPPMAN, *The Complexity of Three-Element Min-Sol and Conservative Min-Cost-Hom*, in Proceedings of the 40th International Colloquium on Automata, Languages, and Programming (ICALP'13), vol. 7965 of Lecture Notes in Computer Science, Springer, 2013, pp. 804–815.
- [72] MAGNUS WAHLSTRÖM, *Half-integrality, LP-branching and FPT algorithms*, in Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'14), SIAM, 2014, pp. 1762–1781.
- [73] M. WAINWRIGHT, T. JAAKKOLA, AND A. WILLSKY, *MAP estimation via agreement on trees: message passing and linear programming*, IEEE Transactions on Information Theory, 51 (2005), pp. 3697–3717.
- [74] MARTIN J. WAINWRIGHT AND MICHAEL I. JORDAN, *Graphical models, exponential families, and variational inference*, Foundations and Trends in Machine Learning, 1 (2008), pp. 1–305.
- [75] TOMÁŠ WERNER, *A Linear Programming Approach to Max-Sum Problem: A Review*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 29 (2007), pp. 1165–1179.

- [76] STANISLAV ŽIVNÝ, *The complexity of valued constraint satisfaction problems*, Cognitive Technologies, Springer, 2012.
- [77] STANISLAV ŽIVNÝ, DAVID A. COHEN, AND PETER G. JEAVONS, *The Expressive Power of Binary Submodular Functions*, Discrete Applied Mathematics, 157 (2009), pp. 3347–3358.

Appendix A. STP Multimorphisms Imply Submodularity.

In this section, we consider symmetric tournament pair (STP) multimorphisms [11] mentioned in Section 3.2.

DEFINITION 4. (a) A pair of operations $\langle \sqcap, \sqcup \rangle$ with $\sqcap, \sqcup : D^2 \rightarrow D$ is called a symmetric tournament pair (STP) if

$$a \sqcap b = b \sqcap a, \quad a \sqcup b = b \sqcup a \quad \forall a, b \in D \quad (\text{commutativity}) \quad (\text{A.1a})$$

$$\{a \sqcap b, a \sqcup b\} = \{a, b\} \quad \forall a, b \in D \quad (\text{conservativity}) \quad (\text{A.1b})$$

(b) Pair $\langle \sqcap, \sqcup \rangle$ is called a submodularity operation if there exists a total order on D for which $a \sqcap b = \min\{a, b\}$, $a \sqcup b = \max\{a, b\}$ for all $a, b \in D$.

(c) Language Γ admits $\langle \sqcap, \sqcup \rangle$ (or $\langle \sqcap, \sqcup \rangle$ is a multimorphism of Γ) if every function $f \in \Gamma$ of arity n satisfies

$$f(x \sqcap y) + f(x \sqcup y) \leq f(x) + f(y) \quad \forall x, y \in D^n$$

It has been shown in [11] that if Γ admits an STP multimorphism then VCSP(Γ) can be solved in polynomial time. STP multimorphisms also appeared in the dichotomy result of [50]:

THEOREM 20. Suppose a finite-valued language Γ is conservative, i.e. it contains all possible unary cost functions $u : D \rightarrow \{0, 1\}$. Then Γ either admits an STP multimorphism or it is NP-hard.

In this paper we prove the following.

THEOREM 21. If a finite-valued language Γ admits an STP multimorphism then it also admits a submodularity multimorphism.

This fact is already known; in particular, footnote 2 in [50] mentions that this result is implicitly contained in [11], and sketches a proof strategy. However, to our knowledge a formal proof has never appeared in the literature. This paper fills this gap. Our proof is different from the one suggested in [50], and inspired some of the proof techniques used in the main part of this paper.

A.1. Proof of Theorem 21. Consider a directed graph $G = (D, E)$. We say that G is a *tournament* if for each pair of distinct labels $a, b \in D$ exactly one of the edges (a, b) , (b, a) belongs to E . We define a one-to-one correspondence between STP multimorphisms $\langle \sqcap, \sqcup \rangle$ and tournaments $G = (D, E)$ as follows:

$$(a, b) \in E \quad \Leftrightarrow \quad (a \sqcap b, a \sqcup b) = (a, b) \quad \forall a, b \in D, a \neq b$$

It can be seen that $\langle \sqcap, \sqcup \rangle$ is a submodularity multimorphism if and only if the corresponding graph G is acyclic.

LEMMA 22. Suppose a finite-valued language Γ admits an STP multimorphism $\langle \sqcap, \sqcup \rangle$ corresponding to a tournament $G = (D, E)$, and suppose that G has a 3-cycle: $(a, b), (b, c), (c, a) \in E$. Let \hat{G} be the graph obtained from G by reversing the orientation of edge (a, b) , and let $\langle \hat{\sqcap}, \hat{\sqcup} \rangle$ be the corresponding STP multimorphism. Then Γ admits $\langle \hat{\sqcap}, \hat{\sqcup} \rangle$.

Proof. Let $\langle \wedge, \vee \rangle$ be the following multimorphism:

$$(x \wedge y, x \vee y) = \begin{cases} (x, y) & \text{if } (x, y) \in \{(a, b), (b, a)\} \\ (x \sqcap y, x \sqcup y) & \text{if } (x, y) \notin \{(a, b), (b, a)\} \end{cases}$$

First, we will prove that Γ admits $\langle \wedge, \vee \rangle$ (step 1), and then prove that Γ admits $\langle \hat{\wedge}, \hat{\vee} \rangle$ (step 2). We fix below function $f \in \Gamma$ of arity n and labellings $x, y \in D^n$.

Step 1 Let us define labellings $x', y' \in D^n$ via

$$(x'_v, y'_v) = \begin{cases} (x_v, x_v \sqcap y_v) & \text{if } (x_v, y_v) \neq (b, a) \\ (c, c) & \text{if } (x_v, y_v) = (b, a) \end{cases} \quad \forall v \in [n]$$

It can be checked that the following identities hold:

$$x' \sqcap y = y' \quad x \sqcup y' = x' \quad (\text{A.2a})$$

$$x \sqcap y' = x \wedge y \quad x' \sqcup y = x \vee y \quad (\text{A.2b})$$

Let us write multimorphism inequalities for pairs (x', y) and (x, y') :

$$\underline{f(x' \sqcap y)} + f(x' \sqcup y) \leq \underline{f(x')} + f(y) \quad (\text{A.3a})$$

$$f(x \sqcap y') + \underline{f(x \sqcup y')} \leq f(x) + \underline{f(y')} \quad (\text{A.3b})$$

Summing (A.3a) and (A.3b), cancelling terms using (A.2a), and then substituting expressions using (A.2b) gives

$$f(x \wedge y) + f(x \vee y) \leq f(x) + f(y) \quad (\text{A.4})$$

Step 2 Let us define labellings $x', y' \in D^n$ via

$$(x'_v, y'_v) = \begin{cases} (x_v \wedge y_v, y_v) & \text{if } (x_v, y_v) \neq (a, b) \\ (c, c) & \text{if } (x_v, y_v) = (a, b) \end{cases} \quad \forall v \in [n]$$

It can be checked that the following identities hold:

$$x' \vee y = y' \quad x \wedge y' = x' \quad (\text{A.5a})$$

$$x \vee y' = x \hat{\vee} y \quad x' \wedge y = x \hat{\wedge} y \quad (\text{A.5b})$$

Let us write multimorphism inequalities for pairs (x', y) and (x, y') :

$$f(x' \wedge y) + \underline{f(x' \vee y)} \leq \underline{f(x')} + f(y) \quad (\text{A.6a})$$

$$\underline{f(x \wedge y')} + f(x \vee y') \leq f(x) + \underline{f(y')} \quad (\text{A.6b})$$

Summing (A.6a) and (A.6b), cancelling terms using (A.5a), and then substituting expressions using (A.5b) gives

$$f(x \hat{\wedge} y) + f(x \hat{\vee} y) \leq f(x) + f(y) \quad (\text{A.7})$$

□

We call the operation of reversing the orientation of edge $(a, b) \in E$ in a graph $G = (D, E)$ a *valid flip* if (a, b) belongs to a 3-cycle. To prove Theorem 21, it thus suffices to show the following:

- For any tournament G there exists a sequence of valid flips that makes it acyclic.

Such sequence can be constructed as follows: (1) start with a subset $B \subseteq D$ with $|B| = 3$; (2) perform valid flips in $G[B]$ to make it acyclic, where $G[B] = (B, E[B])$ is the subgraph of G induced by B ; (3) if $B \neq D$, add a vertex $c \in D - B$ to B and repeat step 2. The lemma below shows how to implement step 2.

LEMMA 23. *Suppose that $G = (B', E)$ is a tournament, $B' = B \cup \{c\}$ with $c \notin B$ and subgraph $G[B]$ is acyclic. Then there exists a sequence of valid flips that makes G acyclic.*

Proof. Suppose that G has a cycle \mathcal{C} , then it must pass through c (since $G[B]$ is acyclic): $\mathcal{C} = \dots \rightarrow b \rightarrow c \rightarrow a \rightarrow \dots$. Since there is a path from a to b in $G[B]$, we must have $(a, b) \in E$ (again, due to acyclicity of $G[B]$). Thus, $c \rightarrow a \rightarrow b \rightarrow c$ is a 3-cycle in G .

Let us repeat the following procedure while possible: pick such cycle and flip edge (c, a) to (a, c) . This operation decreases the number of edges in G coming out of c . Therefore, it must terminate after a finite number of steps and yield an acyclic graph G . \square