

Chapter 2

Expressibility of Valued Constraints

In mathematics you don't understand things. You just get used to them.

John von Neumann

2.1 Introduction

It has been known for some time that the expressive power of crisp constraints is determined by algebraic operations called polymorphisms. Moreover, there is a Galois connection between the set of crisp constraints and the set of operations. This connection has been successfully used in the last decade for complexity analysis of crisp constraint languages [42, 46].

This chapter presents a recently developed algebraic theory for the complexity of valued constraint languages [63, 68], which builds on [65, 274].

2.2 Results

In Sect. 2.3, we survey what is known about the expressive power of crisp constraints, describe the concept of the indicator problem and introduce the concept of a Galois connection. In Sect. 2.4, we show that the question of whether a given cost function is expressible over a finite language is decidable [65]. In Sect. 2.5, we present dual results for fractional operations [274]. Putting results from Sects. 2.4 and 2.5 together, Sect. 2.6 presents a Galois connection between sets of valued constraints and sets of algebraic closure operations [68].

2.3 Indicator Problem

In this section, we discuss the well-known result that the expressive power of crisp constraints is characterised by certain algebraic operations called polymorphisms. We present the construction of an indicator problem, a universal construction for determining whether a given relation is expressible over a crisp constraint language,

and also for determining all polymorphisms of a crisp constraint language. Finally, we show that there is a Galois connection between the set of relations and the set of operations.

Recall Theorem 1.2, which states that the expressive power of a crisp constraint language is fully characterised by its polymorphisms [27, 127, 163]. In other words, for a relation R and a crisp constraint language Γ , the following holds:

$$R \in \langle \Gamma \rangle \iff \text{Pol}(\Gamma) \subseteq \text{Pol}(\{R\}).$$

Remark 2.1 The “ \Rightarrow ” implication follows easily from the fact that expressibility preserves polymorphisms.

This result was obtained by showing that, for any crisp language (that is, set of relations), there is a *universal construction* that can be used to determine whether a relation is expressible in that language, as we now demonstrate.

Definition 2.1 (Indicator Problem) Let Γ be a crisp constraint language over D . For any natural number n , we define the *indicator problem* for Γ of order n as the CSP instance $\mathcal{IP}(\Gamma, n)$ with the set of variables D^n , each with domain D , and constraints $\{C_i\}_{1 \leq i \leq q}$, where $q = \sum_{R \in \Gamma} |R|^n$. For each $R \in \Gamma$, and for each sequence t_1, t_2, \dots, t_n of tuples from R , there is a constraint $C_i = \langle s_i, R \rangle$ with $s_i = \langle v_1, v_2, \dots, v_m \rangle$, where m is the arity of R , and $v_j = \langle t_1[j], t_2[j], \dots, t_n[j] \rangle$, $1 \leq j \leq m$.

Note that, for any crisp constraint language Γ over D , $\mathcal{IP}(\Gamma, n)$ has $|D|^n$ variables, and each corresponds to an n -tuple over D . A concrete example of an indicator problem is given below, and more examples can be found in [164, 166].

Observation 2.1 It is not hard to see from Definitions 2.1 and 1.6 that the solutions to $\mathcal{IP}(\Gamma, n)$ are the polymorphisms of Γ of arity n [165].

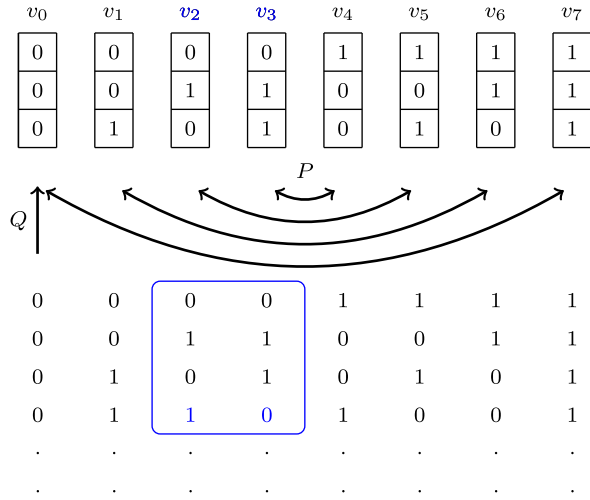
Combining Observation 2.1 with Theorem 1.2 gives:

Corollary 2.1 Let Γ be a crisp constraint language over D . Furthermore, let $R = \{t_1, t_2, \dots, t_n\}$ be a relation over D of arity m . Then R is expressible over Γ , $R \in \langle \Gamma \rangle$, if, and only if, R is equal to the solutions to $\mathcal{IP}(\Gamma, n)$ restricted to the variables v_1, v_2, \dots, v_m , where $v_j = \langle t_1[j], t_2[j], \dots, t_n[j] \rangle$, $1 \leq j \leq m$.

Note that the choice of variables v_1, v_2, \dots, v_m might not be unique as different orderings of the tuples of R can result in different lists. We sketch the proof of Corollary 2.1 since it contains the idea behind the proof of Theorem 1.2.

Proof (Sketch) From Definition 2.1, if R is equal to the solutions to $\mathcal{IP}(\Gamma, n)$ restricted to some subset of variables, then R is expressible over Γ . On the other hand, assume that $R \in \langle \Gamma \rangle$, and denote by \tilde{R} the set of solutions to $\mathcal{IP}(\Gamma, n)$ restricted to

Fig. 2.1 $\mathcal{IP}(\Gamma, 3)$
(Example 2.1)



the variables v_1, v_2, \dots, v_m . It is enough to show that $R = \bar{R}$. By Observation 1.1, all projections are polymorphisms of all relations. Hence, by Observation 2.1, all projections of arity n are solutions of $\mathcal{IP}(\Gamma, n)$. Therefore, $R \subseteq \bar{R}$ from the choice of variables v_1, v_2, \dots, v_m . If $R \neq \bar{R}$, then there must be a solution s to $\mathcal{IP}(\Gamma, n)$ whose restriction to v_1, v_2, \dots, v_m is not contained in R . By Observation 2.1, all solutions to $\mathcal{IP}(\Gamma, n)$ are polymorphisms of Γ , and so is s . But by Remark 2.1, the polymorphism s should be a polymorphism of R , which is a contradiction if $R \neq \bar{R}$. \square

Remark 2.2 Richard Gault implemented a solver called POLYANNA¹ for the indicator problem [126]. An interesting research problem is to investigate various symmetries in the indicator problem and try to make use of them in order to solve instances of the indicator problem more efficiently. However, there is a known worst-case lower bound on the size of any expressibility gadget [270].

Example 2.1 Let $\Gamma = \{P, Q\}$ be a constraint language over $D = \{0, 1\}$, where $P = \{\langle 0, 1 \rangle, \langle 1, 0 \rangle\}$ and $Q = \{\langle 0 \rangle\}$. Given relation $R = \{\langle 0, 0 \rangle, \langle 0, 1 \rangle, \langle 1, 1 \rangle\}$, the task is to determine whether R is expressible over Γ .

Since R consists of three tuples, we construct the indicator problem $\mathcal{IP}(\Gamma, 3)$ of order 3. The variables of $\mathcal{IP}(\Gamma, 3)$ are all 3-tuples over D . There are eight 3-tuples over D , and we denote them by v_0 to v_7 (cf. Fig. 2.1). Since the relation P is binary, any two variables v_i and v_j , which represent the tuples $\langle v_{i1}, v_{i2}, v_{i3} \rangle$ and $\langle v_{j1}, v_{j2}, v_{j3} \rangle$, respectively, are constrained by P if, and only if, all three tuples $\langle v_{i1}, v_{j1} \rangle$, $\langle v_{i2}, v_{j2} \rangle$, and $\langle v_{i3}, v_{j3} \rangle$ belong to P . In our case the following pairs of variables are constrained by P : $\langle v_0, v_7 \rangle$, $\langle v_1, v_6 \rangle$, $\langle v_2, v_5 \rangle$, and $\langle v_3, v_4 \rangle$. The relation

¹<http://www.cs.ox.ac.uk/activities/constraints/software/index.html>.

Q is unary and consists of just one tuple $\langle 0 \rangle$. Therefore, only the variable v_0 , which represents the tuple $\langle 0, 0, 0 \rangle$, is constrained by Q . The construction is illustrated in Fig. 2.1.

Now consider the tuples represented by the variables v_2 and v_3 . These are $\langle 0, 1, 0 \rangle$ and $\langle 0, 1, 1 \rangle$, respectively. If you take these two tuples as columns of a matrix, then the rows of this matrix contain precisely the tuples from R , that is, $\langle 0, 0 \rangle$, $\langle 0, 1 \rangle$, and $\langle 1, 1 \rangle$. However, projecting the solutions to $\mathcal{IP}(\Gamma, 3)$ onto variables v_2 and v_3 does not give R , as the tuple $\langle 1, 0 \rangle$ does not belong to R . (Some of the solutions to $\mathcal{IP}(\Gamma, 3)$ are shown in Fig. 2.1.) Hence R is not expressible over Γ . (Note that we could also obtain the same result by choosing, for instance, variables v_4 and v_6 .)

Recall from Notation 1.8 that, for a crisp constraint language Γ , we denote by $\text{Pol}(\Gamma)$ the set of all polymorphisms of Γ , that is,

$$\text{Pol}(\Gamma) = \{f \mid \forall R \in \Gamma, f \text{ is a polymorphism of } R\}.$$

Notation 2.1 For a set of operations O , we use $\text{Inv}(O)$ to denote the set of relations having all operations in O as polymorphisms, that is,

$$\text{Inv}(\Gamma) = \{R \mid \forall f \in O, f \text{ is a polymorphism of } R\}.$$

Observation 2.2 The result of Theorem 1.2 can be equivalently stated as follows: for any crisp constraint language Γ , it holds that $\langle \Gamma \rangle = \text{Inv}(\text{Pol}(\Gamma))$.

Definition 2.2 (Galois connection [101]) A *Galois connection* between two sets A and B is a pair $\langle F, G \rangle$ of mappings between the power sets $\mathcal{P}(A)$ and $\mathcal{P}(B)$, $F : \mathcal{P}(A) \rightarrow \mathcal{P}(B)$ and $G : \mathcal{P}(B) \rightarrow \mathcal{P}(A)$, such that for all $X, X' \subseteq A$ and all $Y, Y' \subseteq B$ the following conditions are satisfied: $X \subseteq X' \Rightarrow F(X) \supseteq F(X')$ and $Y \subseteq Y' \Rightarrow G(Y) \supseteq G(Y')$; $X \subseteq G(F(X))$ and $Y \subseteq F(G(Y))$.

Notation 2.2 For any finite domain D , we denote by \mathbf{R}_D the set of all relations over D , and we denote by \mathbf{O}_D the set of all operations over D .

The following easy result shows that the $\text{Pol}(\cdot)$ and $\text{Inv}(\cdot)$ operators give rise to an instance of a *Galois connection* between \mathbf{R}_D and \mathbf{O}_D for any finite domain D .

Proposition 2.1 ([234]) *If Γ is a set of relations over D and O is a set of operations over D , then*

1. $O_1 \subseteq O_2 \subseteq \mathbf{O}_D \Rightarrow \text{Inv}(O_1) \supseteq \text{Inv}(O_2)$.
2. $\Gamma_1 \subseteq \Gamma_2 \subseteq \mathbf{R}_D \Rightarrow \text{Pol}(\Gamma_1) \supseteq \text{Pol}(\Gamma_2)$.
3. $\Gamma \subseteq \text{Inv}(\text{Pol}(\Gamma))$.
4. $O \subseteq \text{Pol}(\text{Inv}(O))$.

In order for a Galois connection to be interesting, one should be able to characterise the closures under the two operators. In particular, we are interested in the closure of a set of relations and in the closure of a set of operations under the $\text{Pol}(\cdot)$ and $\text{Inv}(\cdot)$ operators.

Recall that for a crisp constraint language $\Gamma \subseteq \mathbf{R}_D$, we denote by $\langle \Gamma \rangle$ the set of relations that are expressible over Γ . The set $\langle \Gamma \rangle$ is also known as a *relational clone* [234]. For a set of operations $F \subseteq \mathbf{O}_D$, we denote by $\langle F \rangle$ the set of operations from F closed under superposition (also known as composition)² and containing all projections (cf. Observation 1.1). The set $\langle F \rangle$ is known as a *clone of operations*, or just a *clone* [234].

A characterisation of this Galois connection for finite sets D is given by the following two theorems, originally obtained for sets of relations [27, 127].

Theorem 2.1 *For any finite set D , and any $\Gamma \subseteq \mathbf{R}_D$, $\text{Inv}(\text{Pol}(\Gamma)) = \langle \Gamma \rangle$.*

Theorem 2.2 *For any finite set D , and any $F \subseteq \mathbf{O}_D$, $\text{Pol}(\text{Inv}(F)) = \langle F \rangle$.*

The situation is summarised in Fig. 2.2. As with any Galois connection [31], this means that there is a one-to-one correspondence between clones and relational clones. This result shows that the expressive power of any crisp constraint language Γ on a finite set D corresponds to a particular clone of operations on D . Hence the search for tractable crisp constraint languages corresponds to a search for suitable clones of operations [42, 163]. This key observation paved the way for applying deep results from universal algebra in the search for tractable constraint languages [11, 13, 14, 16, 38, 39, 41, 45].

Post completely described the lattice of relational clones and clones over a two-element domain [235]. This description has been heavily used recently to obtain dichotomy complexity classifications of various classes of problems in computer science and artificial intelligence that can be modelled over Boolean domains. For more details, see [28, 29]. More on clone theory can be found in [101, 234].

2.4 Weighted Indicator Problem

In this section, we show that there is also a universal construction to determine whether a given cost function is expressible over a valued constraint language. We briefly describe the result that the expressive power of valued constraints is determined by certain algebraic operations called fractional polymorphisms.

Consider the following problem: given a cost function ϕ of arity m over a domain D , is ϕ expressible over a valued constraint language Γ ? We show that this problem is decidable for every finite Γ . First we prove an upper bound on the number of extra variables needed to express ϕ over Γ .

²Let $f : D^k \rightarrow D$ and $g_1, \dots, g_k : D^\ell \rightarrow D$. The superposition of f and g_1, \dots, g_k is the ℓ -ary operation $f[g_1, \dots, g_k](x_1, \dots, x_\ell) = f(g_1(x_1, \dots, x_\ell), \dots, g_k(x_1, \dots, x_\ell))$.

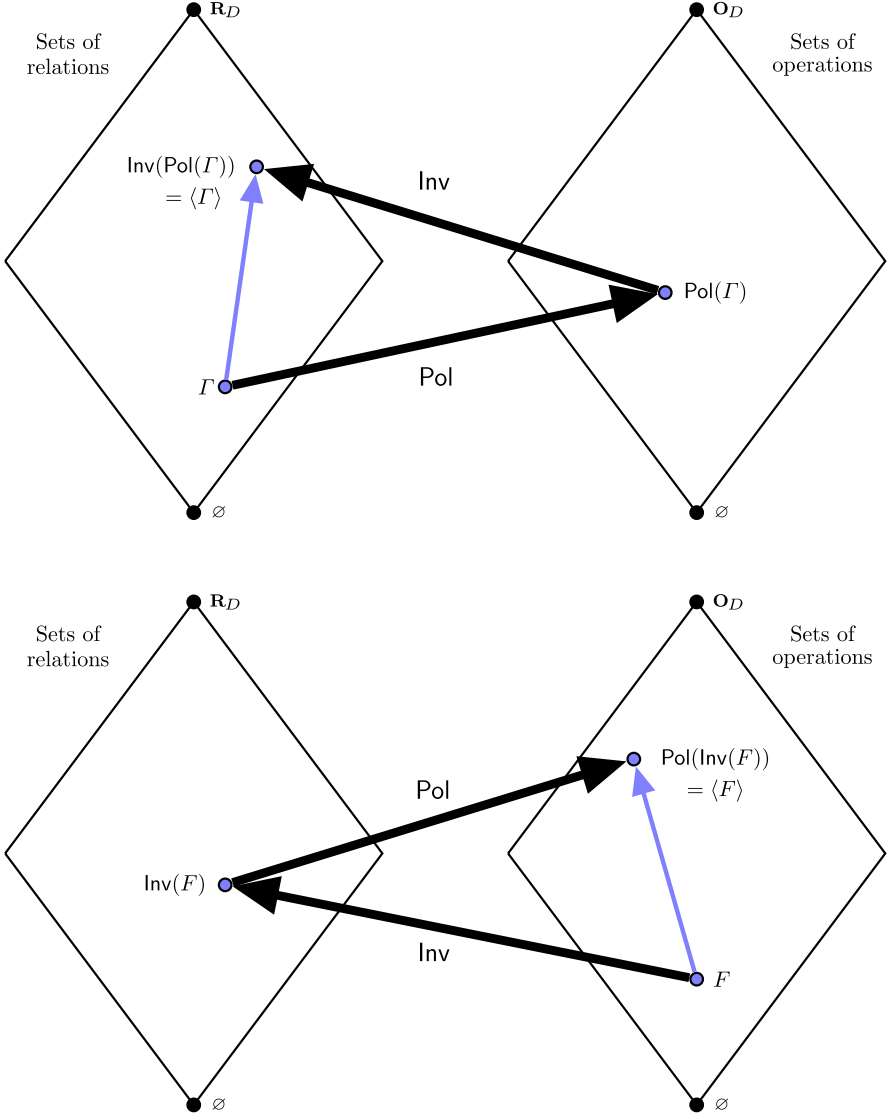


Fig. 2.2 Galois connection between \mathbf{R}_D and \mathbf{O}_D

Proposition 2.2 ([65]) *If a cost function $\phi : D^m \rightarrow \overline{\mathbb{Q}}_{\geq 0}$ is expressible over Γ , then ϕ is expressible over Γ using at most $|D|^{|D|^m}$ hidden variables.*

Proof If $\phi \in \langle \Gamma \rangle$, then by Definition 1.4, there is a gadget $\langle \mathcal{P}, \mathbf{v} \rangle$, where $\mathbf{v} = \langle v_1, \dots, v_m \rangle$, for expressing ϕ over Γ . For the gadget $\langle \mathcal{P}, \mathbf{v} \rangle$ to express ϕ , it has to define ϕ on each of the $|D|^m$ different assignments to \mathbf{v} . Let each of these $|D|^m$ assignments be extended to a complete assignment to all variables of \mathcal{P} (including

hidden variables) in a way that minimises the total cost. For each hidden variable v of $\langle \mathcal{P}, \mathbf{v} \rangle$, we can use the list of $|D|^m$ values assigned to v by these complete assignments to label the variable v . If there are more than $|D|^{|D|^m}$ hidden variables, then two of them will receive the same label. However, this implies that one of the two is redundant, as all constraints involving that variable can replace it with the other variable without changing the overall cost. Hence we require at most $|D|^{|D|^m}$ distinct hidden variables to express ϕ . \square

From this bound on the number of extra variables in a gadget for ϕ over Γ we obtain a decidability result. The idea is to try all possible constraints on all possible subsets of variables, and use linear programming to determine whether there is a combination of these constraints which works.

Theorem 2.3 ([65]) *For a given finite valued constraint language Γ , and a cost function ϕ defined over D , the question of whether ϕ is expressible over Γ is decidable.*

Proof (Sketch) In order to simplify the presentation, we assume that ϕ is a finite-valued cost function. We show how to determine whether there is a gadget for ϕ over Γ , that is, whether there is a VCSP(Γ) instance $\mathcal{P} = \langle V, D, \mathcal{C} \rangle$ and a tuple of variables \mathbf{v} such that $\phi = \pi_{\mathbf{v}}(\mathcal{P})$. By Proposition 2.2, \mathcal{P} has at most $K = |D|^{|D|^m}$ extra variables, where m is the arity of ϕ . Let V be the set of K variables, each associated with a different $|D|^m$ -tuple of values from D . Let E be the $|D|^m \times K$ matrix whose columns are all possible $|D|^m$ -tuples of values from D (or equivalently, variables from V). Observe that there is a $|D|^m \times m$ submatrix E' of E consisting of m columns of E such that the rows of E' correspond to all possible m -tuples of values from D . We let \mathbf{v} be the list of variables corresponding to the columns of E' .

Let \mathcal{A} be the set of all assignments of values from D to the variables from V . Clearly, $|\mathcal{A}| = |D|^K$. We choose $|D|^m$ assignments from \mathcal{A} that correspond to the rows of the matrix E and denote them \mathcal{A}' .

Let $\rho \in \Gamma$ be a cost function of arity k . For an assignment $s \in \mathcal{A}$ and a list of variables \mathbf{u} of length k , recall from Definition 1.1 that we denote by $\rho(s(\mathbf{u}))$ the value of ρ on the list of variables \mathbf{u} assigned by s .

The idea is that if ϕ is expressible over Γ , then there is a set of constraints \mathcal{C} such that $\phi = \pi_{\mathbf{v}}(\mathcal{P})$, where $\mathcal{P} = \langle V, D, \mathcal{C} \rangle$. It remains to show what the set of constraints \mathcal{C} is. And this is where linear programming plays its crucial role.

Let \mathcal{C} be the list of all possible constraints from Γ applied to variables from V . In other words, $\mathcal{C} = \langle C_1, \dots, C_q \rangle$ is an arbitrary but fixed order of the following finite set:

$$\{ \langle \mathbf{u}, \rho \rangle \mid \rho \in \Gamma \text{ of arity } k, \text{ and } \mathbf{u} \text{ is a list of } k \text{ variables from } V \}.$$

We write $C_i = \langle \mathbf{u}_i, \rho_i \rangle$. Clearly,

$$q = \sum_{\rho \in \Gamma \text{ of arity } k} K^k.$$

We define a system of linear equations and inequalities as follows.

For each $s \in \mathcal{A} \setminus \mathcal{A}'$,

$$\sum_{i=1}^q x_i \rho_i(s(\mathbf{u}_i)) \geq \phi(s(\mathbf{v})) + x_0.$$

For each $s \in \mathcal{A}'$,

$$\sum_{i=1}^q x_i \rho_i(s(\mathbf{u}_i)) = \phi(s(\mathbf{v})) + x_0.$$

Note that the variable x_i represents whether the constraint C_i is used in the gadget \mathcal{P} : if $x_i = 0$, then the constraint C_i is not used; if $x_i > 0$, then x_i gives the multiplicity of the constraint C_i . The variable x_0 represents an additive constant up to which the gadget expresses ϕ .

From the construction of the system, ϕ is expressible over Γ if, and only if, there is a nonnegative solution to this system, which is decidable [256]; see also [7]. \square

Remark 2.3 Theorem 2.3 can be extended from finite-valued cost functions to general-valued cost functions [65]. The construction sketched above is known as the *weighted indicator problem*.

Example 2.2 Let $\Gamma = \{\mu, \psi\}$ be the valued constraint language consisting of two cost functions defined over the Boolean domain $D = \{0, 1\}$ as follows:

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

and

$$\psi(x, y) \stackrel{\text{def}}{=} \begin{cases} -1 & \text{if } x = y = 1, \\ 0 & \text{otherwise.} \end{cases}$$

Let ϕ be the ternary cost function defined as follows:

$$\phi(x, y, z) \stackrel{\text{def}}{=} \begin{cases} -1 & \text{if } x = y = z = 1, \\ 0 & \text{otherwise.} \end{cases}$$

The question is whether ϕ is expressible over Γ , that is, whether $\phi \in \langle \Gamma \rangle$. In order to answer this question, we build an instance of the weighted indicator problem as described in the sketched proof of Theorem 2.3. The arity m of ϕ is 3, and hence if ϕ is expressible over Γ , then ϕ is expressible with at most $K = |D|^{|D|^m} = 2^{2^3} = 2^8 = 256$ variables, by Proposition 2.2. Each variable is uniquely identified by an 8-tuple of values from $\{0, 1\}$. We denote by V the set of all such variables with the domain $\{0, 1\}$.

We denote by $\mathbf{v} = \langle v_1, v_2, v_3 \rangle$ the list of three variables, whose corresponding 8-tuples are $t_1 = \langle 0, 0, 0, 0, 1, 1, 1, 1 \rangle$, $t_2 = \langle 0, 0, 1, 1, 0, 0, 1, 1 \rangle$, and $t_3 = \langle 0, 1, 0, 1, 0, 1, 0, 1 \rangle$ respectively. Consider the matrix whose columns are tuples t_1 ,

t_2 , and t_3 . The rows of this matrix are all possible 3-tuples over $\{0, 1\}$. The intuition is that we try to find a gadget \mathcal{P} for ϕ over Γ that expresses ϕ on the variables v_1 , v_2 , and v_3 , that is, $\phi = \pi_{(v_1, v_2, v_3)}(\mathcal{P})$.

Let E be an 8×256 matrix whose columns are the tuples corresponding to variables from V in some fixed order.

We denote by \mathcal{A}' the set of eight assignments of variables from V that are defined by the rows of the matrix E . The intuition is that for every possible assignment s of the variables v_1 , v_2 , and v_3 , we are looking for an assignment s' in \mathcal{A}' which agrees with s (on v_1 , v_2 , and v_3) and the cost of s' is equal to $\phi(v_1, v_2, v_3)$ (up to an additive constant). We denote by \mathcal{A} all assignments of variables from V . Clearly, $|\mathcal{A}| = 2^{256}$.

Now we want to add all possible constraints involving cost functions from Γ . The unary cost function μ can be applied to any of the 256 variables. The binary cost function ψ can be applied to any pair of (not necessarily distinct) variables. Since ψ is symmetric, this gives $\binom{256}{2} + 256$ constraints. In total, we get $2 * 256 + \binom{256}{2} = 33,152$ constraints. Hence we have 33,152 variables x_i that represent whether the i th constraint is used ($x_i > 0$) or not ($x_i = 0$); in the former case, the value of x_i represents the multiplicity of the i th constraint. We then can build a system of linear equations and inequalities as described in the sketch of the proof of Theorem 2.3.

In this particular case, it is not difficult to find a solution to the system of linear equations and inequalities described above. Let y be the variable corresponding to the 8-tuple $\langle 0, 0, 0, 0, 0, 0, 0, 1 \rangle$. We claim that assigning the value 2 to the constraint $\langle \langle y \rangle, \mu \rangle$ (represented by a variable in our system), assigning the value 1 to the constraints $\langle \langle y, x_1 \rangle, \psi \rangle$, $\langle \langle y, x_2 \rangle, \psi \rangle$, and $\langle \langle y, x_3 \rangle, \psi \rangle$, and finally assigning the value 0 to the additive constant $x_0 = 0$ and all other variables is a solution to our system. For any assignment of the variables v_1 , v_2 , and v_3 , setting y to 0 results in total cost 0. If all v_1 , v_2 , and v_3 are assigned 1, setting y to 1 results in total cost -1 . For any other assignment of v_1 , v_2 , and v_3 , setting y to 1 results in total cost ≥ 0 . This corresponds exactly to the definition of the cost function ϕ . This solution gives a gadget for expressing ϕ over Γ using only one extra variable.

Recall Theorem 1.3, which states that the expressive power of a valued constraint language satisfying certain conditions is fully characterised by its polymorphisms and fractional polymorphisms [65]. In other words, for a cost function ϕ and a valued constraint language Γ , such that Γ contains a constant function and is closed under scaling and the feasibility operator, the following holds:

$$\phi \in \langle \Gamma \rangle \quad \Leftrightarrow \quad \text{Pol}(\Gamma) \subseteq \text{Pol}(\{\phi\}) \wedge \text{fPol}(\Gamma) \subseteq \text{fPol}(\{\phi\}).$$

Remark 2.4 The “ \Rightarrow ” implication follows easily from the fact that expressibility preserves polymorphisms and fractional polymorphisms [65]; see also [67].

Remark 2.5 We remark on the assumptions of Theorem 1.3. Notice that it is not a restrictive assumption that every valued constraint language Γ contains a constant function and is closed under scaling. In practice, this corresponds to adding a finite constant that does not alter the relative costs, and to taking more copies of the

same constraint. Therefore, this does not change the complexity of solving VCSP instances over Γ .

We now discuss why it is necessary to assume that Γ is closed under the feasibility operator (or, equivalently, closed under scaling by 0; cf. Remark 1.10) in order to prove equivalence in Theorem 1.3. If \mathcal{F} is a fractional polymorphism of Γ , then \mathcal{F} is also a fractional polymorphism of $\text{Feas}(\Gamma)$. And clearly, any polymorphism of $\text{Feas}(\Gamma)$ is a polymorphism of Γ . Hence for any valued constraint language Γ , $\text{Pol}(\Gamma) \subseteq \text{Pol}(\text{Feas}(\Gamma))$ and $\text{fPol}(\Gamma) \subseteq \text{fPol}(\text{Feas}(\Gamma))$. But this is true for any Γ independently of whether or not $\text{Feas}(\Gamma) \in \langle \Gamma \rangle$; so every valued constraint language Γ satisfies the right-hand side of the equivalence in Theorem 1.3 for $\text{Feas}(\Gamma)$ (that is, $\text{Pol}(\Gamma) \subseteq \text{Pol}(\text{Feas}(\Gamma))$ and $\text{fPol}(\Gamma) \subseteq \text{fPol}(\text{Feas}(\Gamma))$), but not every valued constraint language Γ satisfies $\text{Feas}(\Gamma) \in \langle \Gamma \rangle$.

Fractional polymorphisms on their own characterise the expressive power of finite-valued cost functions and, as shown in Theorem 1.2, polymorphisms on their own characterise the expressive power of crisp cost functions that take only zero and infinite costs.

The proof of Theorem 1.3 given in [65] is based on an application of Farkas' Lemma [256] and uses the concept of the weighted indicator problem. For a given ϕ and Γ , as sketched above, a system of linear equations and inequalities is built such that it has a solution if, and only if, ϕ is expressible over Γ . If this system does not have a solution, then Farkas' Lemma guarantees a certificate of unsolvability. Cohen et al. have shown that the certificate of unsolvability can be turned into a fractional polymorphism \mathcal{F} such that $\mathcal{F} \in \text{fPol}(\Gamma)$, but $\mathcal{F} \notin \text{fPol}(\{\phi\})$ [65].

2.5 Fractional Clones

In this section, we consider the dual question to the one considered in Sect. 2.4: given a finite set Ω of fractional operations over a fixed domain D and a fractional operation \mathcal{F} defined over D , determine whether or not \mathcal{F} belongs to $\text{fPol}(\text{Imp}(\Omega))$. We call this problem the *fractional clone membership* problem.

Using linear programming, we show that this problem is decidable.

Theorem 2.4 *The fractional clone membership problem is decidable.*

Proof Let \mathcal{F} be a k -ary fractional operation $\{\langle r_1, f_1 \rangle, \dots, \langle r_n, f_n \rangle\}$ such that each f_i is a distinct function from D^k to D , each r_i is a positive rational number, and $\sum_{i=1}^n r_i = k$. Let $\Omega = \{\mathcal{F}_1, \dots, \mathcal{F}_q\}$.

Now $\mathcal{F} \notin \text{fPol}(\text{Imp}(\Omega))$ if, and only if, there is a finite-valued cost function ϕ such that $\mathcal{F}_i \in \text{fPol}(\{\phi\})$ for every $1 \leq i \leq q$, but $\mathcal{F} \notin \text{fPol}(\{\phi\})$.

First we show that if there is such a ϕ (we call it a separating cost function), then there is a ϕ of arity at most $m = |D|^k$. Assume that ϕ is of arity strictly larger than m . As there are exactly m different k -tuples over D , any tableau (recall Fig. 1.5)

showing that $\mathcal{F}_i \in \text{fPol}(\{\phi\})$, $1 \leq i \leq q$, and $\mathcal{F} \notin \text{fPol}(\{\phi\})$ has at least one column that occurs twice. However, this column can be removed and the arity of ϕ decreased by 1 by identifying the two arguments corresponding to the two columns. Clearly, if there is a separating cost function ϕ of arity strictly smaller than m , then there is a separating cost function of arity exactly m : we just add dummy variables. Hence we can assume that ϕ is of arity exactly m .

Now we can turn the question of the existence of a separating cost function into a system of linear inequalities. We are looking for $|D|^m$ values (costs of ϕ on all possible assignments) such that for all $1 \leq i \leq q$, $\mathcal{F}_i \in \text{fPol}(\{\phi\})$, and $\mathcal{F} \notin \text{fPol}(\{\phi\})$. But this is easy as showing that $\mathcal{F}_i \in \text{fPol}(\{\phi\})$ is just a question of satisfying a system of linear inequalities for all possible tableaux, by Definition 1.10. Similarly, $\mathcal{F} \notin \text{fPol}(\{\phi\})$ can be expressed as one linear inequality corresponding to the tableau with m different k -tuples over D and the inequality sign the opposite of that in Definition 1.10. This finishes the proof, as the question of whether there is a solution to a system of linear inequalities is decidable [256]. \square

The following example illustrates the technique described in the proof of Theorem 2.4.

Example 2.3 Let $\mathcal{F}_1 = \{\langle 1, \text{Min} \rangle, \langle 1, \text{Max} \rangle\}$, $\mathcal{F} = \{\langle 2, \text{Max} \rangle\}$, and $D = \{0, 1\}$. In order to determine whether $\mathcal{F} \in \text{fPol}(\text{Imp}(\mathcal{F}_1))$, we build a system of linear inequalities as in the proof of Theorem 2.4. We look for a separating cost function ϕ of arity $m = |D|^2 = 4$. Hence we have the $|D|^4 = 16$ variables $x_{0000}, x_{0001}, \dots, x_{1111}$ corresponding to the values of ϕ on 16 different assignments. There are $16 * 16 = 256$ inequalities that make sure that $\{\langle 1, \text{Min} \rangle, \langle 1, \text{Max} \rangle\} \in \text{fPol}(\{\phi\})$:

$$x_{ijkl} + x_{mnop} \geq x_{abcd} + x_{uvyz},$$

where $a = \min(i, m)$, $b = \min(j, n)$, $c = \min(k, o)$, $d = \min(l, p)$, and $u = \max(i, m)$, $v = \max(j, n)$, $y = \max(k, o)$, $z = \max(l, p)$.

Another inequality makes sure that $\{\langle 2, \text{Max} \rangle\} \notin \text{fPol}(\{\phi\})$. According to the proof of Theorem 2.4, the tableau consists of four 2-tuples over $\{0, 1\}$. Hence, the required inequality is

$$x_{0011} + x_{0101} < x_{0111} + x_{0111},$$

where $T = \begin{pmatrix} 0011 \\ 0101 \end{pmatrix}$ on the left-hand side corresponds to four different 2-tuples (columnwise), and $\begin{pmatrix} 0111 \\ 0111 \end{pmatrix}$ on the right-hand side is the application of O to T .

One solution to this system is $x_{00..} = 0$ and $x_{01..} = x_{10..} = x_{11..} = 1$. Notice that ϕ is effectively binary as it only depends on its first two arguments: $\phi(x, y, \cdot, \cdot) = 0$ if $x = y = 0$, and $\phi(x, y, \cdot, \cdot) = 1$ otherwise. It is straightforward to check that this is indeed a solution to the system; that is, $\{\langle 1, \text{Min} \rangle, \langle 1, \text{Max} \rangle\} \in \text{fPol}(\{\phi\})$, but $\{\langle 2, \text{Max} \rangle\} \notin \text{fPol}(\{\phi\})$.

2.6 Galois Theory

We have seen in Theorem 1.3 that the expressive power of languages is determined by the polymorphisms and fractional polymorphisms of the language. However, in order to obtain a Galois connection similar to the one presented for crisp languages in Sect. 2.3, we will have to generalise slightly fractional polymorphisms. The main idea is the following: in the tableau in Fig. 1.5, the upper part corresponds to projections and the bottom part to operations. We relax the definition so that (in some cases) both parts can be arbitrary operations.

Recall that a clone of operations, C , is a set of operations on some fixed set D that contains all projections and is closed under superposition. The k -ary operations in a clone C will be denoted by $C^{(k)}$.

Definition 2.3 (Weighting) We define a k -ary *weighting* of a clone C to be a function $\omega : C^{(k)} \rightarrow \mathbb{Q}$ such that $\omega(f) < 0$ only if f is a projection and

$$\sum_{f \in C^{(k)}} \omega(f) = 0.$$

We denote by \mathbf{W}_C the set of all possible weightings of C and by $\mathbf{W}_C^{(k)}$ the set of k -ary weightings of C .

For any weighting ω , we denote by $\mathbf{dom}(\omega)$ the set of operations on which ω is defined.

Since a weighting is simply a rational-valued function satisfying certain inequalities it can be scaled by any nonnegative rational to obtain a new weighting. Similarly, any two weightings of the same clone of the same arity can be added to obtain a new weighting of that clone.

The notion of superposition for operations can also be extended to weightings in a natural way, as follows.

Definition 2.4 For any clone C , any $\omega \in \mathbf{W}_C^{(k)}$ and any $g_1, g_2, \dots, g_k \in C^{(l)}$, we define the *superposition* of ω and g_1, \dots, g_k , to be the weighting $\omega[g_1, \dots, g_k] \in \mathbf{W}_C^{(l)}$ defined by

$$\omega[g_1, \dots, g_k](f') \stackrel{\text{def}}{=} \sum_{\substack{f \in C^{(k)} \\ f' = f[g_1, \dots, g_k]}} \omega(f). \quad (2.1)$$

Example 2.4 Let C be a clone on some ordered set D and let Max and Min be binary maximum and minimum operations contained in C . Note that $C^{(4)}$ contains operations such as $\text{Max}[e_i^{(4)}, e_j^{(4)}]$, which returns the maximum of the i th and j th argument values. Operations of this form will be denoted by $\text{Max}(x_i, x_j)$.

Let ω be the 4-ary weighting of C given by

$$\omega(f) \stackrel{\text{def}}{=} \begin{cases} -1 & \text{if } f \text{ is a projection, i.e. } f \in \{e_1^{(4)}, e_2^{(4)}, e_3^{(4)}, e_4^{(4)}\}, \\ +1 & \text{if } f \in \{\text{Max}(x_1, x_2), \text{Min}(x_1, x_2), \text{Max}(x_3, x_4), \text{Min}(x_3, x_4)\}, \\ 0 & \text{otherwise,} \end{cases}$$

and let

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

Then, by Definition 2.4 we have

$$\omega[g_1, g_2, g_3, g_4](f) = \begin{cases} -1 & \text{if } f \text{ is a projection, i.e., } f \in \{e_1^{(3)}, e_2^{(3)}, e_3^{(3)}\}, \\ +1 & \text{if } f \in \left\{ \begin{array}{l} \text{Max}(x_1, x_2, x_3), \text{Min}(x_1, x_2), \\ \text{Min}(x_3, \text{Max}(x_1, x_2)) \end{array} \right\}, \\ 0 & \text{otherwise.} \end{cases}$$

Note that $\omega[g_1, g_2, g_3, g_4]$ satisfies the conditions of Definition 2.3 and hence is a weighting of C .

Example 2.5 Let C and ω be the same as in Example 2.4, but now consider

$$\langle g'_1, g'_2, g'_3, g'_4 \rangle = \langle e_1^{(4)}, \text{Max}(x_2, x_3), \text{Min}(x_2, x_3), e_4^{(4)} \rangle.$$

By Definition 2.4 we have

$$\omega[g'_1, g'_2, g'_3, g'_4](f) = \begin{cases} -1 & \text{if } f \in \{e_1^{(4)}, \text{Max}(x_2, x_3), \text{Min}(x_2, x_3), e_4^{(4)}\}, \\ +1 & \text{if } f \in \left\{ \begin{array}{l} \text{Max}(x_1, x_2, x_3), \text{Min}(x_2, x_3, x_4), \\ \text{Min}(x_1, \text{Max}(x_2, x_3)), \\ \text{Max}(\text{Min}(x_2, x_3), x_4) \end{array} \right\}, \\ 0 & \text{otherwise.} \end{cases}$$

Note that $\omega[g'_1, g'_2, g'_3, g'_4]$ does not satisfy the conditions of Definition 2.3 because, for example, we have $\omega[g'_1, g'_2, g'_3, g'_4](f) < 0$ when $f = \text{Max}(x_2, x_3)$, which is not a projection. Hence $\omega[g'_1, g'_2, g'_3, g'_4]$ is not a valid weighting of C .

Definition 2.5 If the result of a superposition is a valid weighting, then that superposition will be called a *proper* superposition.

Remark 2.6 The superposition of a projection operation and other projection operations is always a projection operation. So, by Definition 2.4, for any clone C and any $\omega \in \mathbf{W}_C^{(k)}$, if $g_1, \dots, g_k \in C^{(l)}$ are projections, then the function $\omega[g_1, \dots, g_k]$ can take negative values only on projections, and hence is a valid weighting. This means that a superposition with any list of projections is always a proper superposition.

We are now ready to define *weighted clones*.

Definition 2.6 A *weighted clone*, W , is a nonempty set of weightings of some fixed clone C that is closed under nonnegative scaling, addition of weightings of equal arity, and proper superposition with operations from C . The clone C is called the *support* of W .

Example 2.6 For any clone, C , the set \mathbf{W}_C containing all possible weightings of C is a weighted clone with support C .

Example 2.7 For any clone, C , the set \mathbf{W}_C^0 containing all *zero-valued* weightings of C is a weighted clone with support C .

We now establish a link between weightings and cost functions, which will allow us to link weighted clones and languages closed under expressibility.

Definition 2.7 (Weighted Polymorphism) Let $\phi : D^m \rightarrow \overline{\mathbb{Q}}_{\geq 0}$ be an m -ary cost function on some set D and let ω be a k -ary weighting of some clone of operations C on the set D . We say that ω is a *weighted polymorphism* of ϕ if, for any $\mathbf{x}_1, \dots, \mathbf{x}_k \in D^r$ such that $\phi(\mathbf{x}_i) < \infty$ for $i = 1, \dots, k$, we have

$$\sum_{f \in C^{(k)}} \omega(f) \phi(f(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k)) \leq 0. \quad (2.2)$$

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

Note that if ϕ is improved by the weighting $\omega \in \mathbf{W}_C^{(k)}$, then every element of $C^{(k)}$ must be a polymorphism of ϕ . Thus weighted polymorphisms capture both polymorphisms and fractional polymorphisms.

Example 2.8 The submodular multimorphism $\langle \text{Min}, \text{Max} \rangle$ is equivalent to the 2-ary weighted polymorphism ω defined by

$$\omega(f) \stackrel{\text{def}}{=} \begin{cases} -1 & \text{if } f \in \{e_1^{(2)}, e_2^{(2)}\}, \\ 0 & \text{otherwise.} \end{cases}$$

Remark 2.7 A fractional operation, defined in Definition 1.9 (see also Remark 1.17), is a weighting, defined in Definition 2.3, that assigns weight -1 to all projections. We now show that fractional operations and weightings are the same. Consequently, weighted polymorphisms are the same as fractional polymorphisms. Given a k -ary weighting ω , let

$$\mathcal{F} = \{ \langle \omega(f), f \rangle \mid f \in \mathbf{dom}(\omega) \wedge f \neq e_i^{(k)}, 1 \leq i \leq k \}.$$

Let $c_i = |\omega(e_i^{(k)})|$ be the absolute value of the weight given to the i th projection, and define $c_i = 0$ if $e_i^{(k)} \notin \mathbf{dom}(\omega)$. Let $c = \max_{1 \leq i \leq k} c_i$. If $c = 0$, then we define \mathcal{F} to be $\{ \langle +1, e_i^{(k)} \rangle \}$. Otherwise, we add projections to \mathcal{F} as follows: we add $\langle c + c_i, e_i^{(k)} \rangle$ to \mathcal{F} , if $c + c_i > 0$, for all $1 \leq i \leq k$. Finally, we divide all weights in \mathcal{F} by c .

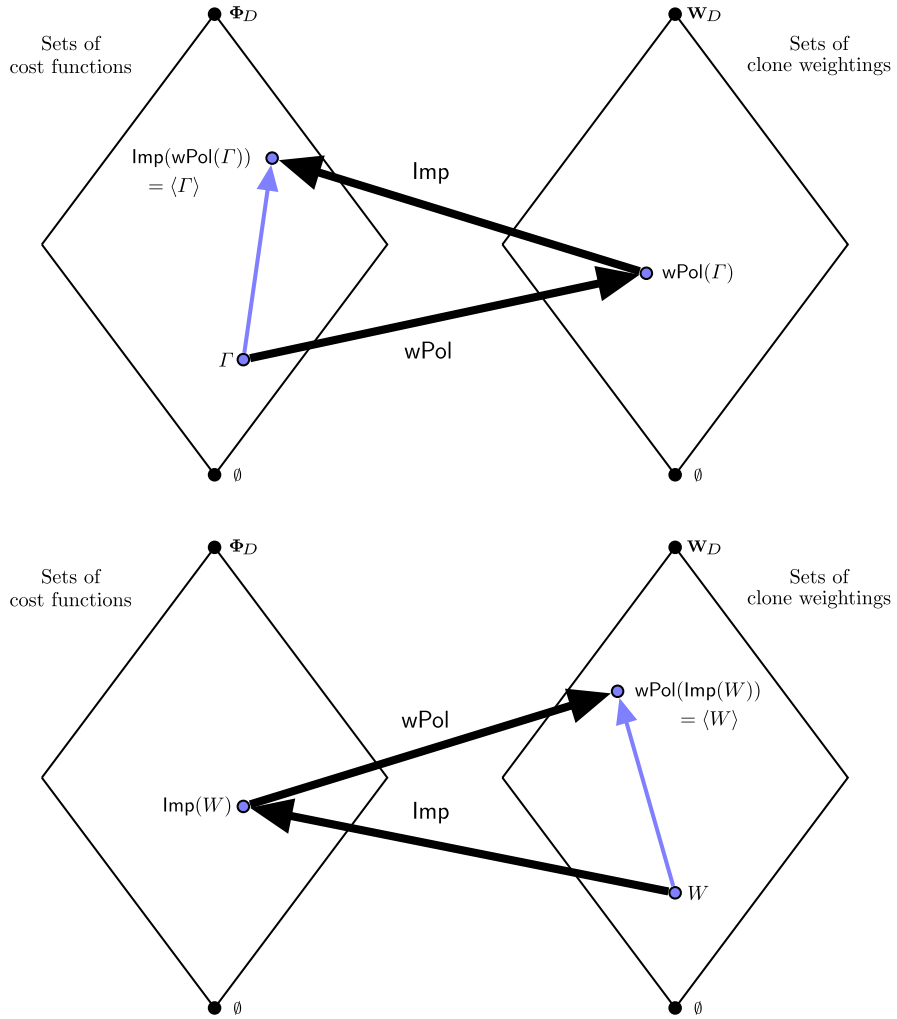


Fig. 2.3 Galois connection between Φ_D and \mathbf{W}_D

Notation 2.3 We denote by Φ_D the set of all cost functions defined on the set D .

Definition 2.8 For any $\Gamma \subseteq \Phi_D$, we denote by $\text{wPol}(\Gamma)$ the set of all weightings of $\text{Pol}(\Gamma)$ that are weighted polymorphisms of all cost functions $\phi \in \Gamma$.

To define a mapping in the other direction, we need to consider the union of the sets \mathbf{W}_C over all clones C on some fixed set D , which will be denoted by \mathbf{W}_D . If we have a set $W \subseteq \mathbf{W}_D$ that may contain weightings of *different* clones over D , then we can extend each of these weightings with zeros, as necessary, so that they are

weightings of the same clone C , given by

$$C = \text{Clone}\left(\bigcup_{\omega \in W} \text{dom}(\omega)\right).$$

This set of extended weightings obtained from W will be denoted by \overline{W} . For any set $W \subseteq \mathbf{W}_D$, we define $\langle W \rangle$ to be the smallest weighted clone containing \overline{W} .

Definition 2.9 For any $W \subseteq \mathbf{W}_D$, we denote by $\text{Imp}(W)$ the set of all cost functions in Φ_D that are improved by all weightings $\omega \in W$.

It follows immediately from the definition of a Galois connection [31] that, for any set D , the mappings $\text{wPol}(\cdot)$ and $\text{Imp}(\cdot)$ form a Galois connection between \mathbf{W}_D and Φ_D , as illustrated in Fig. 2.3. A characterisation of this Galois connection for finite sets D is given by the following two theorems [68].

Theorem 2.5 For any finite set D , and any finite $\Gamma \subseteq \Phi_D$, $\text{Imp}(\text{wPol}(\Gamma)) = \langle \Gamma \rangle$.

Theorem 2.6 For any finite set D , and any finite $W \subseteq \mathbf{W}_D$, $\text{wPol}(\text{Imp}(W)) = \langle W \rangle$.

As with any Galois connection [31], this means that there is a one-to-one correspondence between languages closed under expressibility and weighted clones. Hence, the search for tractable languages over a finite set corresponds to a search for suitable weighted clones of operations.

The proofs of Theorems 2.6 and 2.5 rely on the application of Farkas' Lemma [256] and are based on the ideas presented in Sects. 2.4 and 2.5, respectively.

Creed and Živný have used the algebraic theory to classify so-called minimal Boolean languages [85], thus obtaining (a simpler proof of) the hardness part of the complexity classification of Boolean languages [67] (cf. Chap. 6), where the original results from [67] relied on ad hoc gadgets.

2.7 Summary

We have investigated the expressive power of crisp and valued constraints. We have presented a construction to determine whether a given cost function is expressible over a given language. We have also presented a construction to determine whether a given fractional polymorphism belongs to a fractional clone. We have then presented a general algebraic theory. This algebraic theory allowed for a simpler proof of the (hardness part of the) classification of Boolean languages [85], and also a simpler proof of the (hardness part of the) classification of conservative languages [86], originally obtained in [187, 188] (more in Chap. 7).

2.7.1 Related Work

Galois connections for various variants of the CSP have been considered in the literature [31, 255].

2.7.2 Open Problems

The most interesting open problem is the structure of weighted clones. Due to the presented Galois connection, this would immediately shed some light on the complexity of VCSPs.

The Complexity of Valued Constraint Satisfaction Problems

Zivny, S.

2012, XVIII, 170 p., Hardcover

ISBN: 978-3-642-33973-8