A note on some collapse results of valued constraints

Bruno Zanuttini [a,1]

[a] *GREYC, Université de Caen Basse-Normandie, Boulevard du Maréchal Juin, 14 032 Caen Cedex, France*

Stanislav Živný [b,2,*]

[b] *Computing Laboratory, University of Oxford, Wolfson Building, Parks Road, Oxford OX1 3QD, UK*

Abstract

Valued constraint satisfaction problem (VCSP) is an optimisation framework originally coming from Artificial Intelligence and generalising the classical constraint satisfaction problem (CSP). The VCSP is powerful enough to describe many important classes of problems. In order to investigate the complexity and expressive power of valued constraints, a number of algebraic tools have been developed in the literature. In this note we present alternative proofs of some known results without using the algebraic approach, but by representing valued constraints explicitly by combinations of other valued constraints.

Key words:
Valued constraint satisfaction problems, Expressive power, Max-closed constraints, Theory of computation

1. Introduction

The Valued Constraint Satisfaction Problem (VCSP) [18] is a general framework to model various optimisation problems [1,17], which generalises the classical constraint satisfaction problem (CSP) [15,7,8,17].

Informally, in the VCSP framework, an instance consists of a set of variables, a set of possible values, and a set of (soft) constraints. Each constraint has an associated cost function which assigns a cost (or a degree of violation) to every possible tuple of values for the variables in the scope of the constraint. The goal is to find an assignment of values to all of the variables which has the minimum total cost. The set of cost functions used in the description of the problem is called the *valued constraint language*. Infinite costs can be used to indicate infeasible solutions, and if the range of all cost functions is $\{0, \infty\}$ (so-called hard constraints), we get the standard CSP framework as a special case.

An important problem is determining which additional constraints can be *expressed* by a given valued constraint language. The notion of *expressibility* has been a key component in the analysis of complexity for the classical CSP model [12,2]. It was also a major tool in the complexity analysis of a wide variety of Boolean constraint problems carried out by Creignou et al. [7], where it was referred to as *implementation*. Expressibility is a particular form of problem reduction: if a constraint can be polynomially expressed in a given constraint language, then it can be added to the language without changing the computational complexity of the associated class of problems. Hence determining what can be expressed in a given valued constraint language is a fundamental step in the complexity analysis of valued constraint problems.

A number of algebraic tools have been developed to understand the complexity and expressive power of constraint languages [12,4,3]. Using the algebraic approach, Cohen et al. showed [5] that some valued constraint languages, including the set of all monotonic valued constraints, can be expressed using only valued constraints of a fixed finite arity. They also showed some other classes of valued constraints, including the set of all monotonic valued constraints with finite cost values, which cannot be expressed by a subset of any fixed finite arity, and hence form an infinite hierarchy [5].

In this paper we consider the class of monotonic, so-called *max-closed* valued constraints. Our main contributions are new proofs of certain recent results on the expressive power of max-closed valued constraints [5]. The proofs presented

[*] Corresponding author.
Email addresses: bruno.zanuttini@info.unicaen.fr (Bruno Zanuttini), stanislav.zivny@comlab.ox.ac.uk (Stanislav Živný).
[1] This work was supported by ANR grant CANAR (ANR-06-BLAN-0383-02).
[2] This work was supported by EPSRC grant EP/F01161X/1.

here use a very different approach: they do not rely on the algebraic approach, but are based on an explicit representation of valued constraints. Although some of the proofs presented here are simple and considered folklore, we also present new and more involved proofs, thus providing a logical approach to the expressive power of VCSP.

VCSP instances with max-closed valued constraints are well known to be solvable in polynomial time (generalising the tractable language for the standard SATISFIABILITY problem consisting of Horn clauses) [13,4], but much less is known about their expressive power. The new proofs presented here have a number of interesting new features:

First, the alternative proofs are simpler, not requiring the machinery of the algebraic approach. Moreover, they are constructive and thus give new techniques for studying valued constraint satisfaction problems from the logical point of view, as has proven useful in the propositional case [7].

Second, recent progress on the expressive power of submodular valued constraint has shown that both the algebraic approach and the non-algebraic approach, presented in this paper, can play important roles in investigating the expressive power of valued constraints. In particular, a combination of these two approaches has proved particularly effective.

Submodular constraints are a key concept in operational research and combinatorial optimisation, see for example [16,20,19]. As submodular valued constraints can be characterised as min-max-closed valued constraints [4], understanding the expressive power of max-closed valued constraints is a natural first step towards complete understanding of the expressive power of submodular valued constraints.

Moreover, the techniques presented in this paper have been recently used successfully. First, using explicit representations of submodular valued constraints, Živný and Jeavons showed [23] a new class of submodular valued constraints of arbitrary arities expressible by binary submodular valued constraints, thus describing a new class of VCSP instances which can be solved efficiently using MIN-CUT. This class was independently discovered in [21]. Next, using a combination of the explicit representation of submodular valued constraints and certain algebraic results, Živný et al. showed [22] a novel class of submodular valued constraints which can be solved efficiently using MIN-CUT. Furthermore, they also showed that there are submodular valued constraints which *cannot* be expressed by binary submodular valued constraints [22], thus answering an open problem which has been considered within several different contexts in computer science, including computer vision, artificial intelligence, and pseudo-Boolean optimisation. This provides an example how important it is to understand various aspects of the expressive power of valued constraints. We believe that understanding both the algebraic and non-algebraic aspects of the expressive power of valued constraints can contribute to the development of better algorithms for discrete optimisation problems.

2. Background

In the original definition of the VALUED CONSTRAINT SATISFACTION PROBLEM (VCSP) given in [18], costs were allowed to lie in any positive totally ordered monoid called a *valuation structure*. For our purposes it is sufficient to consider costs which lie in the set $\overline{\mathbb{N}}$ consisting of all natural numbers (including zero) together with infinity [3].

Given some fixed set D, a function from D^k to $\overline{\mathbb{N}}$ will be called a *cost function*. If the range of ϕ lies within $\{0, \infty\}$, then ϕ is called a *crisp* cost function. Note that crisp cost functions correspond precisely to relations. If the range of ϕ lies entirely within $\mathbb{N}$, then ϕ is called a *finite-valued* cost function. Finally, if ϕ takes both finite and infinite values, then ϕ is called a *general* cost function.

Definition 1 *An instance $\mathcal{I}$ of* VCSP *is a triple $\langle V, D, \mathcal{C}\rangle$, where V is a finite set of* variables*, which are to be assigned values from the set D, and $\mathcal{C}$ is a set of* valued constraints*. Each $c \in \mathcal{C}$ is a pair $c = \langle \boldsymbol{v}, \phi\rangle$, where $\boldsymbol{v}$ is a tuple of variables of length m, called the* scope *of c, and $\phi : D^m \to \overline{\mathbb{N}}$ is a cost function. An* assignment *for the instance $\mathcal{I}$ is a mapping s from V to D. We extend s to a mapping from V^k to D^k on tuples of variables by applying s component-wise. We denote by $\mathcal{A}$ the set of all assignments. The* cost *of an assignment s is defined as follows:*

$$Cost_{\mathcal{I}}(s) = \sum_{\langle \boldsymbol{v}, \phi\rangle \in \mathcal{C}} \phi(s(\boldsymbol{v})).$$

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

The VCSP is a very general framework which allows us to describe many optimisation problems, including many NP-hard problems [7,14,8,17]. Various restrictions which give rise to classes of problems solvable in polynomial time have been studied in the literature [4,11].

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

Definition 2 *For any* VCSP *instance $\mathcal{I} = \langle V, D, \mathcal{C}\rangle$, and any tuple $\boldsymbol{v}$ of m variables of $\mathcal{I}$, the* projection *of $\mathcal{I}$ onto $\boldsymbol{v}$, denoted $\pi_{\boldsymbol{v}}(\mathcal{I})$, is the m-ary cost function defined as follows:*

$$\pi_{\boldsymbol{v}}(\mathcal{I})(\boldsymbol{x}) = \min\{Cost_{\mathcal{I}}(s) \mid s \in \mathcal{A}, s(\boldsymbol{v}) = \boldsymbol{x}\}.$$

We say that a cost function ϕ is expressible *over a valued constraint language Γ if there exists an instance $\mathcal{I} \in$* VCSP *such that all cost functions in $\mathcal{I}$ are from Γ and a tuple $\boldsymbol{v}$ of*

[3] See [6] for a discussion of why limiting ourselves to the $\overline{\mathbb{N}}$ valuation structure is not a severe restriction.

variables of $\mathcal{I}$ such that $\pi_{\boldsymbol{v}}(\mathcal{I}) = \phi$. We call the pair $\langle \mathcal{I}, \boldsymbol{v}\rangle$ a representation of ϕ by Γ. [4]

Example 3 *Let ϕ be the ternary cost function which returns the sum of its arguments. For instance, the assignment defined by $s(v_1) = 1, s(v_2) = 2, s(v_3) = 2$ has cost $\phi(\langle 1, 2, 2\rangle) = 1 + 2 + 2 = 5$. Now let $\mathcal{I} = \langle V, D, \mathcal{C}\rangle$ with $V = \{v_1, v_2, v_3\}$, $D = \{1, 2, 3\}$ and $\mathcal{C}$ contains the constraint $\langle\langle v_1, v_2, v_3\rangle, \phi\rangle$ and the (crisp) constraint $v_1 \neq v_3$. The projection of $\mathcal{I}$ onto $\boldsymbol{v} = \langle v_1, v_2\rangle$ is the binary cost function such that $\pi_{\boldsymbol{v}}(\mathcal{I})(x_1, x_2) = x_1 + x_2 + 1$ if $x_1 \neq 1$, and $\pi_{\boldsymbol{v}}(\mathcal{I})(1, x_2) = x_2 + 3$. The intuition is that if $x_1 \neq 1$, then the assignment to v_1, v_2 can be completed with $v_3 = 1$, and otherwise it can be completed with $v_3 = 2$ (since 1 violates $v_1 \neq v_3$ and thus yields an infinite cost, which is bigger than the finite cost obtained by assigning $v_3 = 2$).*

Note that the notion of expressibility for crisp cost functions (=relations) corresponds to the standard notion of expressibility using conjunction and existential quantification (*primitive positive formulas*) [2]. We denote by $\langle\Gamma\rangle$ the *expressive power* of Γ which is the set of all cost functions expressible over Γ up to additive and multiplicative constants. [5]

From now on, we assume D is totally ordered with some relation $<$, and we use $<$ to define a partial order on tuples component-wise. We also use the function max component-wise on tuples.

Definition 4 *A cost function $\phi : D^k \to \overline{\mathbb{N}}$ is called* max-closed *if for every $u, v \in D^k$, $2\phi(\max(u, v)) \leq \phi(u) + \phi(v)$.*

The class of max-closed relations was first introduced in [13] and shown to be tractable. In other words, any VCSP instance with max-closed relations over a finite set was shown to be polynomial-time solvable. The class of general and finite-valued max-closed cost functions was introduced and shown to be tractable in [4].

Lemma 5 ([4, Lemma 6.14]) *If ϕ is max-closed, then ϕ is finitely antitone, that is, for all tuples u, v with $\phi(u), \phi(v) < \infty$, $u \leq v$ implies $\phi(u) \geq \phi(v)$.*

Definition 6 *For every $d \geq 2$, we define the following:*

- *$\mathbf{R}_{d,m}$ denotes the set of all relations of arity at most m over a domain of size d, and $\mathbf{R}_d = \cup_{m\geq 0} \mathbf{R}_{d,m}$;*
- *$\mathbf{R}^{\max}_{d,m}$ denotes the set of all max-closed relations of arity at most m over an ordered domain of size d, and $\mathbf{R}^{\max}_d = \cup_{m\geq 0} \mathbf{R}^{\max}_{d,m}$;*
- *$\mathbf{F}_{d,m}$ denotes the set of all finite-valued cost functions of arity at most m over a domain of size d, and $\mathbf{F}_d = \cup_{m\geq 0} \mathbf{F}_{d,m}$;*
- *$\mathbf{F}^{\max}_{d,m}$ denotes the set of all finite-valued max-closed cost functions of arity at most m over an ordered domain of size d, and $\mathbf{F}^{\max}_d = \cup_{m\geq 0} \mathbf{F}^{\max}_{d,m}$.*
- *$\mathbf{G}_{d,m}$ denotes the set of all general cost functions of arity at most m over a domain of size d, and $\mathbf{G}_d = \cup_{m\geq 0} \mathbf{G}_{d,m}$;*
- *$\mathbf{G}^{\max}_{d,m}$ denotes the set of all general max-closed cost functions of arity at most m over an ordered domain of size d, and $\mathbf{G}^{\max}_d = \cup_{m\geq 0} \mathbf{G}^{\max}_{d,m}$.*

[4] The pair $\langle \mathcal{I}, \boldsymbol{v}\rangle$ is also known as a *gadget* for expressing ϕ over Γ [5], or a *constraint network* [8], or an *implementation* [7].

[5] This is only due to technical reasons, but it does not change the complexity of the language, that is, L and $\langle L\rangle$ remain polynomial-time equivalent to each other whether additive and multiplicative constants are allowed or not [3].

3. Results

Theorem 7 ([5], alternative proof) *For every $d \geq 3$,*

(i) $\mathbf{R}_2 = \langle \mathbf{R}_{2,3}\rangle$;
(ii) $\mathbf{R}^{\max}_2 = \langle \mathbf{R}^{\max}_{2,3}\rangle$;
(iii) $\mathbf{R}_d = \langle \mathbf{R}_{d,2}\rangle$;
(iv) $\mathbf{R}^{\max}_d = \langle \mathbf{R}^{\max}_{d,2}\rangle$;
(v) $\mathbf{G}^{\max}_d \subseteq \langle \mathbf{R}^{\max}_{d,2} \cup \mathbf{F}^{\max}_{d,1}\rangle$;
(vi) $\mathbf{G}^{\max}_2 \subseteq \langle \mathbf{R}^{\max}_{2,3} \cup \mathbf{F}^{\max}_{2,1}\rangle$.

Theorem 7 was originally proved in [5] using algebraic properties of cost functions: in the case of relations using so-called polymorphisms [2] and in the case of general cost functions using so-called fractional polymorphisms [3]. Here we give a different proof using explicit representations of cost functions which does not rely on these algebraic properties.

PROOF.

(i) It is well known that any relation $R \in \mathbf{R}_2$ can be expressed as a propositional formula ψ in conjunctive normal form: simply a conjunction of clauses which disallow tuples not in R . By the standard SAT to 3-SAT reduction [9], which replaces a clause $(l_1 \vee l_2 \vee \ldots \vee l_k)$ with

$$(l_1 \vee l_2 \vee \neg y_1) \wedge (y_1 \vee l_3 \vee \neg y_2) \wedge \ldots \wedge (y_{k-3} \vee l_{k-1} \vee l_k)$$

using new variables $y_1, \ldots, y_{k-3}$, ψ is equivalent to $\exists y_1, \ldots, y_{k-3}\psi'$ where ψ' consists of clauses of length at most three. Hence $\langle \psi', \langle l_1, \ldots, l_k\rangle\rangle$ is a representation of ψ by $\mathbf{R}_{2,3}$.

(ii) It is known [13,10] that a relation is max-closed if and only if it can be represented by a conjunction of anti-Horn clauses, that is, clauses of the form $(x_1 \geq a_1 \vee \cdots \vee x_k \geq a_k)$ or $(x_1 \geq a_1 \vee \cdots \vee x_k \geq a_k \vee y \leq b)$, where $a_1, \ldots, a_k, b$ are domain values. In the Boolean domain this corresponds to the condition that every clause has at most one negated literal. It is easy to verify that the standard SAT to 3-SAT reduction in the proof of (i) preserves the anti-Horn form of clauses (consider the possibly one negated variable as the last literal). Therefore, the proof of (i) proves (ii) as well.

(iii) Because every relation is logically equivalent to some conjunction of clauses [13,10], the proof of (i) gives a weaker result that $\mathbf{R}_d \subseteq \langle \mathbf{R}_{d,3}\rangle$. We need to show how to express a clause C of length 3 (over the domain D, where $d = |D| \geq 3$) as a conjunction of clauses of length 2. Let $D = \{1, \ldots, d\}$ and $C = (U_1(x_1) \vee U_2(x_2) \vee U_3(x_3))$ for some literals (unary relations) U_i, $1 \leq i \leq 3$. We claim that C is equivalent to $\exists y C' = (U_1(x_1) \vee N_1(y)) \wedge (U_2(x_2) \vee N_2(y)) \wedge (U_3(x_3) \vee N_3(y))$ where y is a new variable and $N_1(y) = D \setminus \{1\}$ ("not 1"), $N_2(y) = D \setminus \{2\}$ and $N_3(y) = \{1, 2\}$. It is not difficult to see that a satisfying assignment of C can be extended to a satisfying assignment of C' and conversely, a satisfying assignment of C' gives a satisfying assignment of C.

(iv) It is enough to show that any clause of the form $(x_1 \geq a_1 \vee \cdots \vee x_k \geq a_k)$ or $(x_1 \geq a_1 \vee \cdots \vee x_k \geq a_k \vee y \leq b)$, where $a_1, \ldots, a_k, b$ are domain values, can be expressed by a conjunction of anti-Horn clauses over at most two variables.

Let C be a clause in $\mathbf{R}_d^{\max}$:

$$C = (x_1 \geq a_1 \vee x_2 \geq a_2 \vee \cdots \vee x_k \geq a_k \vee y \leq b)$$

(the case without the y literal is even easier and can be handled similarly).

For all $i = 1, \ldots, k-1$, let y_i be a fresh variable with d values, so at least three values, say $1, 2, 3$ with the natural order. We define the following conjunction of clauses ψ, where $y_i \in \{1, 3\}$ is used as a shorthand for $y_i \geq 3 \vee y_i \leq 1$ (possible values less than 1 or greater than 3 do not matter) as follows:

$$\begin{aligned}\psi \;=\; & (x_1 \geq a_1 \vee y_1 \in \{1,3\}) \\ & \wedge (y_1 \leq 2 \vee x_2 \geq a_2) \wedge \\ & \wedge \bigwedge_{i=2}^{k-1} \begin{pmatrix} (y_{i-1} \geq 2 \vee y_i \in \{1,3\}) \\ \wedge (y_i \leq 2 \vee x_{i+1} \geq a_{i+1}) \end{pmatrix} \\ & \wedge (y_{k-1} \geq 2 \vee y \leq b).\end{aligned}$$

The intuition is given by reading the second clause as $(y_1 \geq 3 \rightarrow x_2 \geq a_2)$ and the third one as $(y_1 \leq 1 \rightarrow y_2 \in \{1, 3\})$. Since the first clause reads "either $x_1 \geq a_1$ or $y_1 \geq 3$ or $y_1 \leq 1$", together with the above implications this gives "either $x_1 \geq a_1$ or $x_2 \geq a_2$ or $y_2 \in \{1, 3\}$". Iterating this reasoning, one can see intuitively why the construction works.

More formally, we show that C is logically equivalent to $\exists y_1 \ldots y_{k-1} \psi$. First, let t be a tuple satisfying ψ. Then if t satisfies $x_1 \geq a_1$, we are done. Otherwise, because of the first clause in ψ, t must satisfy (1) $y_1 \geq 3$ or (2) $y_1 \leq 1$. In case (1), because of the second clause in ψ, t must satisfy $x_2 \geq a_2$ and we are done. In case (2), because of the third clause in ψ, t must satisfy $y_2 \geq 3 \vee y_2 \leq 1$, and we proceed by induction.

Conversely, let t be a tuple satisfying C. We show that t can be completed into a model of ψ by assignments to the y_i's.

Assume first that t satisfies $x_1 \geq a_1$. Then completing t with $t(y_i) = 2$ for all $i = 1, \ldots, k-1$ yields a model of ψ whatever the values assigned by t to $x_2, \ldots, x_k, y$. This can be seen by examining each clause in ψ.

Now assume that t satisfies $x_{i_0} \geq a_{i_0}$ for some $i_0 \in \{2, \ldots, k\}$. Then completing t with $t(y_i) = 1$ for all $i = 1, \ldots, i_0 - 2$, $t(y_{i_0-1}) = 3$ and $t(y_i) = 2$ for all $i = i_0, \ldots, k-1$ again yields a model of ψ.

Finally, assume that t satisfies $y \leq b$. Then completing t with $t(y_i) = 1$ for all $i = 1, \ldots, k-1$ yields a model of ψ, which finishes the proof.

Note that this proof makes clear why the same argument does not work for $d = 2$. Indeed, the "hole" in literal $y_i \in \{1, 3\}$ is necessary, since otherwise this literal would be tautologous and thus, so would every second clause be in ψ. We showed in (ii) that in the Boolean case, ternary relations are sufficient and it is well known (see [5]) that they are indeed necessary.

(v) Let ϕ be an m-ary general max-closed cost function, and write $x_1, \ldots, x_m$ for the variables. Let $y_1, \ldots, y_K$ be variables (with d values, say $1, \ldots, d$), and let $K = \max\{\phi(\boldsymbol{x}) \mid \phi(\boldsymbol{x}) < \infty\}$ be the biggest finite cost in the range of ϕ. Intuitively, a cost of k for a tuple will be encoded by $y_1, \ldots, y_k$ assigned a cost of 1 (and the others 0).

We first encode infinite costs. Let ϕ_R be the relation $\{u \mid \phi(u) < \infty\}$, that is, the crisp cost function assigning a cost of 0 to exactly these tuples assigned a finite cost by ϕ (and ∞ to the others). It turns out that this relation is max-closed. Indeed, for all $u, v \in \phi_R$ we have $\phi(u), \phi(v) < \infty$ by definition of ϕ_R. Since ϕ is max-closed we have $2\phi(\max(u, v)) \leq \phi(u) + \phi(v) < \infty$, so $\phi(\max(u, v)) < \infty$ and thus, $\max(u, v) \in \phi_R$. So ϕ_R is max-closed, that is, $\phi_R \in \mathbf{R}_d^{\max}$.

We now encode finite costs. For an m-tuple t with $\phi(t) < \infty$, write k_t for $\phi(t)$. We let ψ_t be the anti-Horn formula $\bigwedge_{j=1}^{k_t} ((\bigvee_{i=1}^m x_i > t[i]) \vee y_j \leq 1)$. Observe that this formula reads $\boldsymbol{x} \leq t \rightarrow y_1 \leq 1 \wedge \cdots \wedge y_{k_t} \leq 1$.

Finally, we define the anti-Horn formula ψ to be $\psi_R \wedge \bigwedge_{t \in D^m} \psi_t$, where ψ_R is an anti-Horn formula equivalent to ϕ_R. By (iv), this formula can be expressed over $\mathbf{R}_{d,2}^{\max}$.

The formula ψ encodes the cost of every tuple as a number of y_j's assigned 1. We thus add, to every variable y_j, $j = 1, \ldots, K$, the cost function μ defined by $\mu(1) = 1$ and $\mu(2) = \cdots = \mu(d) = 0$. Clearly, this function is max-closed and therefore in $\mathbf{F}_{d,1}^{\max}$.

We now show that $\langle \psi, \langle x_1, \ldots, x_m \rangle \rangle$ is a representation of ϕ. Let t be an m-ary tuple. Assume first that $k_t = \phi(t)$ is finite. Then ψ contains the subformula

$$\bigwedge_{j=1}^{k_t} ((\bigvee_{i=1}^{m} x_i > t[i]) \vee y_j \leq 1).$$

Since obviously $t[i] > t[i]$ holds for no i, every assignment which satisfies ψ sets variables $y_1, \ldots, y_{k_t}$ to 1 and thus, has a cost of at least k_t. Now let s_t be the assignment which is equal to t over $x_1, \ldots, x_m$ and which assigns 1 to $y_1, \ldots, y_{k_t}$ and 2 to $y_{k+1}, \ldots, y_K$. We show that s_t satisfies ψ, which gives an assignment of cost at most k_t.

First let $\psi_{t'} \in \psi$, and recall that $\psi_{t'}$ reads $\boldsymbol{x} \leq t' \rightarrow y_1 \leq 1 \wedge \cdots \wedge y_{\phi(t')} \leq 1$. If $t \leq t'$, then since ϕ is max-closed and both costs are finite (by definition of $\psi_{t'}$), we have $\phi(t) \geq \phi(t')$ by Lemma 5. It follows $\{y_1, \ldots, y_{\phi(t')}\} \subseteq \{y_1, \ldots, y_{k_t}\}$, so s_t assigns 1 to $y_1, \ldots, y_{\phi(t')}$ and thus satisfies $\psi_{t'}$. Otherwise, if $t \not\leq t'$, then $t[i] > t'[i]$ for some i and thus s_t satisfies $\psi_{t'}$ (it does not satisfy its premises). Finally, s_t satisfies $\psi_{t'}$ for all t'.

Now s_t satisfies ψ_R by definition of ϕ_R, since s_t equals t over $x_1, \ldots, x_m$ and $\phi(t) < \infty$ by assumption. We finally have that for all t with finite cost under ϕ, s_t satisfies ψ and thus, the projection of ψ assigns a cost of at most k_t to t. Since the cost is at least k_t as shown above, we have the result.

Now if t has infinite cost under ϕ, then by definition of ϕ_R we have that t does not satisfy ψ and thus, has infinite cost under the projection of ψ as well, as desired.

(vi) The same as the proof of (v). In this case ψ is a relation over the Boolean domain, and therefore ψ can be expressed, by (ii), over $\mathbf{R}_{2,3}^{\max}$.

Acknowledgements The authors would like to thank the anonymous reviewers for useful comments on an earlier draft of this paper. Stanislav Živný would like to thank Peter Jeavons for many helpful discussions and ongoing support and encouragement.

References

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

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

[3] Cohen, D., Cooper, M., Jeavons, P.: An algebraic characterisation of complexity for valued constraints. In: Proceedings of the 12th International Conference on Principles and Practise of Constraint Programming (CP'06). Volume 4204 of LNCS. (2006) 107–121

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

[5] Cohen, D.A., Jeavons, P.G., Živný, S.: The expressive power of valued constraints: Hierarchies and collapses. Theoretical Computer Science **409**(1) (2008) 137–153

[6] Cooper, M.: High-order consistency in valued constraint satisfaction. Constraints **10** (2005) 283–305

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

[8] Dechter, R.: Constraint Processing. Morgan Kaufmann (2003)

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

[10] Gil, A., Hermann, M., Salzer, G., Zanuttini, B.: Efficient algorithms for description problems over finite totally ordered domains. SIAM Journal on Computing **38**(3) (2008) 922–945

[11] Grohe, M.: The complexity of homomorphism and constraint satisfaction problems seen from the other side. Journal of the ACM **54**(1) (2007)

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

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

[14] Khanna, S., Sudan, M., Trevisan, L., Williamson, D.: The approximability of constraint satisfaction problems. SIAM J. on Computing **30**(6) (2001) 1863–1920

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

[16] Nemhauser, G., Wolsey, L.: Integer and Combinatorial Optimization. John Wiley & Sons (1988)

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

[18] Schiex, T., Fargier, H., Verfaillie, G.: Valued constraint satisfaction problems: hard and easy problems. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95). (1995)

[19] Schrijver, A.: Combinatorial Optimization: Polyhedra and Efficiency. Volume 24 of Algorithms and Combinatorics. Springer-Verlag (2003)

[20] Topkis, D.: Supermodularity and Complementarity. Princeton University Press (1998)

[21] Zalesky, B.: Efficient determination of Gibbs estimators with submodular energy functions. arXiv:math/0304041v1 (February 2008)

[22] Živný, S., Cohen, D.A., Jeavons, P.G.: The Expressive Power of Binary Submodular Functions. Technical report (November 2008) arXiv:0811.1885v1 [cs.DM], Submitted for publication.

[23] Živný, S., Jeavons, P.G.: Classes of Submodular Constraints Expressible by Graph Cuts. In: Proceedings of the 14th International Conference on Principles and Practice of Constraint Programming (CP'08). Volume 5202 of LNCS. (2008) 112–127