Maria Eriksson
Examensarbete 2005
Naturvetarprogrammet 160 p
Department of Information Technology
Uppsala University
Supervisor: Pierre Flener
Examinator: Justin Pearson

Detecting Symmetries in Relational Models of CSPs

Abstract

This master's thesis studies symmetry detection in constraint satisfaction problems (CSPs). After modelling some well-known CSPs in a relational language, the symmetries of the single constraints in the models are studied. These results are then transformed into general patterns for symmetry detection. When the symmetries of the single constraints have been detected, symmetries for complete CSPs, with one or more constraints, are derived compositionally. This work deals with value as well as variable interchangeability.

Sammanfattning

Inom villkorsprogrammering arbetar man med så kallade kombinatoriska problem (CSPs) där målet är att tilldela variabler värden så att vissa givna villkor är uppfyllda. Ett exempel på ett välkänt sådant problem är uppgiften att placera åtta damer på ett schackbräde så att ingen dam hotar någon annan dam. Variablerna är i detta fall de åtta damernas placering och värdena är brädets 64 rutor. De villkor som impliceras av problemet kan t ex uttryckas genom att två damer, vilka som helst, inte får stå i samma rad, samma kolumn eller samma diagonal. En lösning på problemet är en placering av damerna så att dessa villkor är uppfyllda.

En del kombinatoriska problem är dessutom optimeringsproblem (COPs) därigenom att det finns ett numeriskt uttryck vars värde, vilket är beroende av variablernas värden, dessutom ska optimeras (minimeras eller maximeras). Ett enkelt exempel på ett sådant problem är det att tilldela två variabler, x och y, värden mellan 1 och 10 så att uttrycket $x + y$ antar ett så litet värde som möjligt.

I många kombinatoriska problem är olika värden eller variabler sinsemellan utbytbara utan att problemets lösningar påverkas. Låt oss exemplifiera detta med det välkända fyrfärgsproblemet. Uppgiften är att i en graf tilldela varje nod en av fyra färger så att två sammanhängande noder inte får samma färg. Antag nu att vi har en lösning på detta problem. Eftersom det enligt problemformuleringen inte har någon betydelse vilken specifik färg varje nod får utan endast om två noders färger är lika eller inte kan vi i vår lösning byta ut två färger mot varandra och därigenom erhålla en ny lösning, d v s om alla noder som i den första lösningen var "blå" nu blir "röda" och alla "röda" noder blir "blå" så kommer villkoret att två sammanhängande noder inte får ha samma färg fortfarande att vara uppfyllt. Vi säger att denna andra lösning är symmetrisk med den första. I detta fall uppstår symmetrin genom att olika värden är sinsemellan utbytbara, men det kan också vara fallet att variablerna i en modell av ett problem är utbytbara på motsvarande sätt. I en del problem är inte alla, utan endast vissa, värden eller variabler sinsemellan utbytbara och vi kan tala om bitvis utbytbarhet. I många kombinatoriska problem är vi endast intresserade av att hitta en lösning, vi behöver alltså inte leta efter alla symmetriska varianter av denna. Genom att på ett tidigt stadium upptäcka vilka symmetrier som finns i problemet, såsom att i fyrfärgsproblemet se att det inte har någon betydelse vilken specifika färger noderna har, kan vi undvika att leta efter symmetriska lösningar och därigenom väsentligt förkorta tiden det tar att hitta en lösning. När det är känt vilka symmetrier som finns i en modell av ett problem finns det olika strategier för att "bryta" dessa symmetrier så att vi inte behöver leta efter alla symmetriska lösningar. En hel del arbete har lagts ner på att ta fram sådana strategier. För att kunna bryta symmetrierna är det dock nödvändigt att först känna till vilka symmetrier som finns. Att upptäcka symmetrier är därför ett annat viktigt forskningsområde inom villkorsprogrammeringen.

I denna examensuppsats studeras hur symmetrier kan upptäckas i kombina-

toriska (optimerings)problem. Detta har gått till så att några välkända kombinatoriska problem, såsom 8-damer-problemet och fyrfärgsproblemet, har modellerats i ett (relationellt) programmeringsspråk. Därefter har symmetrier hos de enskilda villkoren i varje modell studerats. När symmetrier hos vissa typer av villkor eller delar av villkor på detta sätt har upptäckts manuellt har dessa resultat sedan omvandlats till generella mönster för olika typer av symmetriska villkor. Efter att symmetrier för de enskilda villkoren har upptäckts på detta sätt, kan symmetrierna för hela kombinatoriska problem, med ett eller flera villkor, härledas kompositionellt. Med det menas att vi, när vi känner till symmetrierna hos de enskilda villkoren hos en modell av ett problem, kan härleda symmetrier för hela problemet, vilket i detta fall görs med hjälp av resultat från [4]. Denna uppsats behandlar symmetrier som uppstår såväl på grund av olika värdens utbytbarhet, som utbytbarhet mellan variabler.

Contents

1 Introduction

In many constraint satisfaction problems, symmetries occur, due to the fact that either certain variables or some domain values are indistinguishable. Since often only one solution, instead of many symmetric solutions, to a particular problem is wanted, breaking these symmetries, by introducing some distinction between the variables/values, eg. a lexicographical order, is an important tool for efficiently solving CSPs. But in order to be able to break the symmetries, one already has to know where they occur and the aim of this work is to show how value/variable-symmetries can be detected in relational models of CSPs. The approach is to detect/derive symmetries in single constraints, and then to use the insight from [4] that more complex symmetries can be derived compositionally for problems with more than one constraint.

In Section 2 relational modelling is presented, as well as the constraints used when modelling the problems. Then, in Section 3, some example CSPs are presented and modelled. These are the progressive party, scene allocation, social golfers, graph colouring, BIBD, n-queens and warehouse problems. The models in Figures 4, 6 and 8 are my models and the other models of Section 3 are found in [2] and [4]. In Section 4 is the concept of value symmetry and useful results connected to it presented. All the definitions, propositions and examples of that section are from [4] except Proposition 22 which I have derived from two other propositions of Section 4. In Section 7 I translate the results on *value* symmetry of Section 4 into results on *variable* symmetry. This is done in Definitions, Propositions and Examples 27 to 43. The Definitions and Propositions 26 and 44 to 50 of that section are from [4]. In Section 5 and Section 8 I use the theoretical results on symmetries from Sections 4 and 7 to derive value/variable symmetries for the constraints in the models presented in Section 3. This is done stepwise, by first looking for common patterns for the constraints in the models, for which full symmetries can easily be detected, and then generalising these patterns to handle partial symmetries as well. After that the complete symmetry for each problem can be derived compositionally, as is done in Sections 6 and 9.

2 Relational Modelling

All the problems of Section 3 are modelled in a relational ESRA-like language (for our purposes, the description of ESRA given in [2] suffices, but a complete description of ESRA is given in [1]) and for three of the problems (progressive party, scene allocation and n-Queens problems) models in an OPL-like language are also included. This is to give examples of how the constraints in the ESRA-like models (mainly the count constraint) are related to the well-known global constraints of the OPL-like language, such as the allDifferent constraint. In order to be able to understand the relational models of Section 3, a short explanation of the constraints found in them, and the relation between these constraints and global constraints, will be given here.

The constraints of the relational models of Section 3 are expressed in what

can be characterised as second-order logic with counting. This means that apart from functions, relations and quantifiers, ranging over individuals and relations and for which the syntax and semantics is closely related to second-order logic, we have a generalisation of the existential quantifier, count, with the following syntax:

$$\text{count}(\langle \mathit{Set} \rangle)(\langle \mathit{QuantExpr} \rangle)$$

This states that the number of elements defined by the quantifier expression $\langle \mathit{QuantExpr} \rangle$ is in the set $\langle \mathit{Set} \rangle$.

Example 1 The expression $\text{count}(\{1, 2, 4\})(x : Xs \mid x < 3)$ says that there are 1, 2 or 4 elements x in the set Xs, such that $x < 3$, i.e. this holds for instance if $Xs = \{-2, 0, 1, 2\}$, but not if $Xs = \{0, 1, 2\}$.

As mentioned, count is a generalisation of the existential quantifier, as $\exists(x : Xs \mid \mathit{Condition})$ is equivalent to $\text{count}(1 \ldots \text{sup})(x : Xs \mid \mathit{Condition})$, where sup is positive infinity. In Section 3, the count constraint of the relational models corresponds to different global constraints of the OPL-like models. As we will see in Section 3, a count constraint in a relational model might for example correspond to an allDifferent constraint in an OPL-like model. In this section, we only mention something about the correspondence between a count and an allDifferent constraint. If the set in an allDifferent constraint is extensionally given, we have the following equivalence:

$$\text{allDifferent}(\{v_1, \ldots, v_n\}) \equiv \text{count}(n)(\{v_1, \ldots, v_n\})$$

But if the sets in the allDifferent/count constraints are intensionally given there are several count constraints equivalent to an allDifferent constraint:

$$\begin{aligned}
& \text{allDifferent}(\{v_i : D \mid i : I\}) && (1)\\
\equiv\ & \forall(d : D)\text{count}(0 \ldots 1)(i : I \mid v_i = d) && (2)\\
\equiv\ & \forall(i : I)\text{count}(0)(j \neq i : I \mid v_i = v_j) && (3)\\
\equiv\ & \forall(i : I)\text{count}(1)(j : I \mid v_i = v_j) && (4)\\
\equiv\ & \text{count}(\text{card}(I))(v_i : D \mid i : I) && (5)\\
\equiv\ & \text{count}(\text{card}(I))(d : D \mid \exists i \in I : v_i = d) && (6)
\end{aligned}$$

3 Sample Models

3.1 The Progressive Party Problem

The goal of the progressive party problem is to schedule a party at a yacht club. Some boats are designated as hosts, while the crews of other bots are guests. The guests visit different host boats over a number of periods. There are three constraints for these visits, namely that any guest crew can visit any host boat in at most one period (constraint 1), any two guest crews can visit the same host at the same period at most once (constraint 2) and at any period, the spare

```
range Guests = ...;
range Hosts = ...;
range Periods = ...;
int spareCap[Hosts] = ...;
int crewSize[Guests] = ...;
var Hosts schedule[Guests,Periods];
solve {
  forall(g in Guests)
    allDifferent(all(p in Periods) schedule[g,p]);
  forall(i in Guests, j in Guests: i<j)
    meetAtmost(all(p in Periods) schedule[i,p],
               all(p in Periods) schedule[j,p], 1);
  forall(p in Periods)
    weightedAtmost(crewSize, all(g in Guests) schedule[g,p],
                   spareCap);
};
```

Figure 1: An OPL-like model of the Progressive Party problem

dom $Guests,\ Hosts,\ Periods$
cst $spareCap : Hosts \longrightarrow \mathbb{N}$
cst $crewSize : Guests \longrightarrow \mathbb{N}$
var $schedule : (Guests \times Periods) \longrightarrow Hosts$
solve

$$\forall(g : Guests, h : Hosts)\ \text{count}(0 \ldots 1)(p : Periods \mid schedule(g,p) = h)$$
$$\wedge\ \forall(i < j : Guests)\ \text{count}(0 \ldots 1)(p : Periods \mid schedule(i,p) = schedule(j,p))$$
$$\wedge\ \forall(p : Periods,\ h : Hosts) \left(\sum_{g:Guests \mid schedule(g,p)=h} crewSize(g) \right) \leq spareCap(h)$$

Figure 2: A relational model of the Progressive Party problem

capacity of any host boat must not be exceed of the sum of the crew sizes of all the guest crews visiting that boat at that period (constraint 3). Two different models of this problem are found in Figures 1 and 2. The OPL-like model in Figure 1 is similar to the one found in [4], except that some variables have changed names and that the constraints have a different order, and the ESRA-like model in Figure 2 is found in [2]. If we compare the constraints in these two models with the ones in Section 2 we see that the allDifferent constraints of Figure 1 is of type (1) and that it corresponds to a count constraint of type (2) in Figure 2. We could just as well have used for example a constraint of type (5) in Figure 2 which instead would give us the following constraint:

$$\forall(g : Guests)\text{count}(\text{card}(Periods))(schedule(g,p) \mid p : Periods)$$

```
range Scenes = ...;
range Days = ...;
range Actors = ...;
int fee[Actors] = ...;
{Scenes} casting[Actors] = ...;
var Days shoot[Scenes];
minimise
  sum(a in Actors)
    fee[a]*nbDistinct(all(s in casting[a]) shoot[s])
subject to
  forall(d in Days)
    atMost(all(s in Scenes: shoot[s]=d), 5);
```

Figure 3: An OPL-like model of the Scene Allocation problem

dom $Scenes, Days, Actors$
cst $fee : Actors \longrightarrow \mathbb{N}$
cst $casting : Actors \times Scenes$
var $shoot : Scenes \longrightarrow Days$
minimise
$$\sum_{a:Actors} fee(a)*\text{card}\{(d : Days) \mid \exists(s : Scenes \mid casting(a, s) \wedge shoot(s) = d)\}$$
such that
$$\forall(d : Days)\text{count}(0 \ldots 5)(s : Scenes \mid shoot(s) = d)$$

Figure 4: A relational model of the Scene Allocation problem

3.2 Scene Allocation

The next problem we look at is the scene allocation problem, where the objective is to minimise the cost for shooting a film, where each scene of the film needs the participation of a certain set of actors, who have to be present the whole day that scene is shot, and where each actor has an individual fee for the time spent in the studio. The production cost is to be minimised, by deciding which scene should be shot which day, under the condition that at most five scenes can be shot each day. The problem can be modelled as in Figure 3 and Figure 4 respectively, where the model of Figure 3 is found in [4].

3.3 Social Golfers

Like the progressive party problem, the social golfers problem is a scheduling problem, but here it is n players at a golf club whose playing is to be scheduled for w weeks, such that any two players are scheduled to play in the same group at most once and that every group of players each week consists of s players. A third condition for the problem is that every player plays golf exactly once a week, and this condition is in the model implicitly taken care of by the fact that

cst $g, s, w : \mathbb{N}$
dom $Players = 1 \ldots g * s$
dom $Weeks = 1 \ldots w$
dom $Groups = 1 \ldots g$
var $schedule : (Players \times Weeks) \longrightarrow^{s*w} Groups$
solve
$\forall(p < q : Players)$ count $(0 \ldots 1)(v : Weeks \mid schedule(p, v) = schedule(q, v))$
$\wedge\ \forall(h : Groups, v : Weeks)$ count $(s)(p : Players \mid schedule(p, v) = h)$

Figure 5: A relational model of the Social Golfers problem

dom $Colours$
dom $Nodes$
cst $edge : Nodes \times Nodes$
var $colouring : Nodes \longrightarrow Colours$
solve
$\forall(n : Nodes)\text{count}(0)(m : Nodes \mid edge(n, m) \wedge colouring(n) = colouring(m))$

Figure 6: A relational model of the Graph Colouring problem

functions are total in the ESRA-like language. The relational model of Figure 5 is found in [2].

3.4 Graph Colouring

In the well known graph colouring problem, nodes of a graph are to be given any of a given set of colours, such that any two connected nodes do not have the same colour. A relational model of this problem is found in Figure 6.

3.5 Balanced Incomplete Block Design

A *balanced incomplete block design* (BIBD) is an arrangement of v elements, called *varieties* into b *blocks*, such that each block contains k varieties, and each variety occurs in r different blocks (which in the model of Figure 7 is expressed by the multiplicities of the $\times$ constructor), with the additional condition that each pair of distinct varieties occurs together in exactly λ blocks. A relational model of the BIBD problem, from [2], is found in Figure 7. There we see that whereas the decision variables in our previous models were *functions*, here we are dealing with a *relation*. Also, whereas the problems in our previous models have some degree of *value symmetry*, the BIBD problem modelled in this way is *variable symmetric*, as we will see in Section 8.

dom $\mathit{Varieties}, \mathit{Blocks}$
cst $r, k, \lambda : \mathbb{N}$
var $\mathit{bibd} : \mathit{Varieties}\ {}^{r}\!\times^{k}\ \mathit{Blocks}$
solve
$\quad \forall(v_1 < v_2 : \mathit{Varieties})\ \mathrm{count}(\lambda)(j : \mathit{Blocks} \mid \mathit{bibd}(v_1, j) \wedge \mathit{bibd}(v_2, j))$

Figure 7: A relational model of BIBDs

dom $\mathit{Warehouses}, \mathit{Stores}$
cst $\mathit{maintCost} : \mathbb{N}$
cst $\mathit{cap} : \mathit{Warehouses} \longrightarrow \mathbb{N}$
cst $\mathit{supplyCost} : \mathit{Stores} \times \mathit{Warehouses} \longrightarrow \mathbb{N}$
var $\mathit{supplier} : \mathit{Stores} \longrightarrow \mathit{Warehouses}$
minimise
$$\left(\sum_{w:\mathit{Warehouses}, s:\mathit{Stores} \mid \mathit{supplier}(s)=w} \mathit{supplyCost}(s, w)\right)$$
$$+\mathit{maintCost} * \mathrm{card}\{(w : \mathit{Warehouses}) \mid \exists(s : \mathit{Stores} \mid \mathit{supplier}(s) = w)\}$$
such that
$\quad \forall(w : \mathit{Warehouses})\ \mathrm{count}(0 \ldots \mathit{cap}(w))(s : \mathit{Stores} \mid \mathit{supplier}(s) = w)$

Figure 8: A relational model of the Warehouse Location problem

3.6 The Warehouse Problem

The background to the warehouse problem, which is a constraint optimisation problem, is that some company considers opening warehouses on some candidate locations in order to supply its existing stores. The objective is to decide which warehouses to open, while minimising a cost function. The stores and warehouses have the following properties:

- Each candidate warehouse has the same maintenance cost.
- Each candidate warehouse has a supply capacity (the maximum number of stores it can supply).
- The supply cost to a store depends on the warehouse.

The problem has two constraints:

- Each store must be supplied by exactly one actually opened warehouse.
- Each actually opened warehouse supplies at most a number of stores equal to its capacity.

And the cost function which it is the objective to minimise:

- The sum of the actually incurred maintenance costs and supply costs.

A model of this problem is found in Figure 8.

```
int n = ...;
range Queens = 1 ... n;
range Rows = 1 ... n;
var Rows nqueen[Queens];
solve {
  allDifferent(all(q in Queens) nqueen[q]);
  allDifferent(all(q in Queens) q-nqueen[q]);
  allDifferent(all(q in Queens) q+nqueen[q]);
};
```

Figure 9: An OPL-like model of the n-Queens problem

cst n
dom $Queens = 1 \dots n$
dom $Rows = 1 \dots n$
var $nqueen : Queens \longrightarrow Rows$
solve
$\forall(q : Queens)\ \text{count}(0)(r \neq q : Queens \mid nqueen(q) = nqueen(r))$
$\wedge\ \forall(q : Queens)\ \text{count}(0)(r \neq q : Queens \mid q - nqueen(q) = r - nqueen(r))$
$\wedge\ \forall(q : Queens)\ \text{count}(0)(r \neq q : Queens \mid q + nqueen(q) = r + nqueen(r))$

Figure 10: A relational model of the n-Queens problem

3.7 The n-Queens Problem

The n-Queens problem consists in placing n queens on an $n \times n$ chessboard, so that no two queens attack each other. This problem can be modelled using three constraints, as in Figure 9 and in Figure 10. Whereas the model in Figure 9 uses the global allDifferentconstraint expressing the fact that for any solution to the problem, all queens must be in different diagonals and rows, the relational model in Figure 10 is more closely related to the first description of the problem. In Figure 10 the first count constraint is of type (3) of Section 2, but we could of course just as well have used a constraint of type (4) instead. Note that the constraint that the queens are in different columns is already implicitly expressed in both models, where queen i can be seen as the queen in column i.

4 Value Interchangeability

As we noted in the beginning, symmetries often occur in CSPs, because values or variables are interchangeable. In this section the concept of *(piecewise) value interchangeability* will be presented. This concept is presented in [3] and developed in [4]. We illustrate the concept of value interchangeability with a formulation from [3]: "There are many applications in resource allocation and scheduling where the exact values taken by the variables are not important. What is significant is which variables take the *same* values or, in other terms,

how the variables are clustered."

Apart from Proposition 22 the rest of this section is copied from [4], and the number of the definitions, examples and propositions in that paper is given in parentheses. Definition 2 not only defines a CSP and the solution of a CSP, but also indicates the notation that will be used in the following pages:

Definition 2 A *CSP* is a triple $\langle V, D, C\rangle$, where V denotes the set of variables, D denotes the set of possible values for these variables, and $C : (V \to D) \to Bool$ is a constraint that specifies which assignments of values to the variables are solutions. A *solution* to a CSP $\mathcal{P} = \langle V, D, C\rangle$ is a function $\sigma : V \to D$ such that $C(\sigma) = true$. The set of solutions to a CSP $\mathcal{P}$ is denoted by $Sol(\mathcal{P})$.

Let us apply this definition to the model of the graph colouring problem in Figure 6. There we have a set of nodes, *Nodes* for each of which the function *colouring* should assign a colour. The variables in this model is the set $\{colouring(n) \mid n : Nodes\}$. The domain of the model is the possible values for the variables, i.e. the set *Colours* and the constraint is of course the forall statement, which for each assignment of values to the variables is true of false. A solution to the problem is an assignment of a colour to each varible such that the constraint is true, i.e. such that no two adjacent nodes have the same colour.

Some of the problems of Section 3 are *constraint optimisation problems*, *COPs*. Problems of this type, apart from being CSPs, also have a global function. A solution to the COP $\mathcal{O} = \langle \mathcal{V}, \mathcal{D}, \mathcal{C}, \{\rangle$ is a solution to the CSP $\mathcal{P} = \langle \mathcal{V}, \mathcal{D}, \mathcal{C}\rangle$ such that the value of f is the minimal one among the possible values of f for solutions to $\mathcal{P}$. The scene allocation and the warehouse problems are examples of COPs, where we have some function whose value is to be minimised. In the rest of this section, we will deal with value interchangeability for CSPs and COPs.

Definition 3 (Definition 5) Let $\mathcal{P} = \langle V, D, C\rangle$ be a CSP. $\mathcal{P}$ is *value interchangeable* if, for each solution $\sigma \in Sol(\mathcal{P})$ and each bijection $b : D \to D$, the function $b \circ \sigma \in Sol(\mathcal{P})$.

Example 4 (Example 1) Let $V \supseteq \{v_1, v_2, v_3\}$. The CSP $\mathcal{P} = \langle V, D, C\rangle$, where $C = \text{allDifferent}(v_1, v_2, v_3)$ is value interchangeable.

Definition 5 (Definition 6) Let $\mathcal{D} = \{D_1, \ldots, D_n\}$ be a partition of D. A bijection $b : D \to D$ is *piecewise interchangeable* over $\mathcal{D}$ if $\forall v \in D_i : b(v) \in D_i \;\; (1 \leq i \leq n)$.

We say that a partition $\mathcal{S}_1$ of a set S is *coarser* than the partition $\mathcal{S}_2$ of S, or equivalently, that $\mathcal{S}_2$ is *finer* than $\mathcal{S}_1$, if every element of $\mathcal{S}_2$ is a subset of some element of $\mathcal{S}_1$.

Definition 6 (Definition 7) Let $\mathcal{P} = \langle V, D, C\rangle$ be a CSP and $\mathcal{D}$ be a partition of D. $\mathcal{P}$ is *piecewise value interchangeable* (PVI) over $\mathcal{D}$ if, for each solution $\sigma \in Sol(\mathcal{P})$ and each piecewise-interchangeable bijection b over $\mathcal{D}$, $b \circ \sigma \in Sol(\mathcal{P})$.

Note that, if $\mathcal{P} = \langle V, D, C\rangle$ is value interchangeable, then it is piecewise value interchangeable over $\{D\}$. It might also be the case that a CSP $\mathcal{P} = \langle V, D, C\rangle$ is PVI over the partition $\mathcal{D}_1$, as well as over the partition $\mathcal{D}_2$. Then if $\mathcal{D}_1$ is coarser than $\mathcal{D}_2$, we say that $\mathcal{P}$ being PVI over $\mathcal{D}_1$ is a *stronger* symmetry than $\mathcal{P}$ being PVI over $\mathcal{D}_2$. In this case we say that the latter symmetry is *suboptimal*.

Example 7 (Example 2) Let $V \supseteq \{v_1, v_2, v_3\}$, $D \ni 1$, and consider a constraint $\text{atmost}(o, d, \langle v_1, \ldots, v_k\rangle)$ which holds for an assignment σ if there are at most o occurrences of d in $\langle \sigma(v_1), \ldots, \sigma(v_k)\rangle$. The CSP $\langle V, D, \text{atmost}(2, 1, \langle v_1, v_2, v_3\rangle)\rangle$ is PVI over $\{\{1\}, D \setminus \{1\}\}$.

A CSP might also have global functions, which are value interchangeable if their value does not change under certain bijections of the domain values.

Definition 8 (Definition 8) A global function $f : (V \to D) \to \mathbb{N}$ is *value interchangeable* if, for each assignment $\sigma : V \to D$ and each bijection $b : D \to D$, $f(\sigma) = f(b \circ \sigma)$.

Example 9 (Example 3) Let $V \supseteq \{v_1, v_2, v_3\}$ and consider global functions of the form $\text{nbDistinct}(v_1, \ldots, v_k)$ which, given an assignment σ, return the number of distinct values in $\langle \sigma(v_1), \ldots, \sigma(v_k)\rangle$. The global function $\text{nbDistinct}(v_1, v_2, v_3)$ is value interchangeable.

Definition 10 (Definition 9) Let $\mathcal{D}$ be a partition of D. A global function $f : (V \to D) \to \mathbb{N}$ is *piecewise value interchangeable* over $\mathcal{D}$ if, for each assignment $\sigma : V \to D$ and piecewise-interchangeable bijection b over $\mathcal{D}$, $f(\sigma) = f(b \circ \sigma)$.

Now we can define value interchangeability for a COP as well.

Definition 11 (Definition 10) Let $\mathcal{O} = \langle V, D, C, f\rangle$ be a COP. $\mathcal{O}$ is *value interchangeable* if, for each solution $\sigma \in Sol(\mathcal{O})$ and each bijection $b : D \to D$, $b \circ \sigma \in Sol(\mathcal{O})$ and $f(\sigma) = f(b \circ \sigma)$.

Definition 12 (Definition 11) Let $\mathcal{O} = \langle V, D, C, f\rangle$ be a COP and $\mathcal{D}$ be a partition of D. $\mathcal{O}$ is *piecewise value interchangeable* over $\mathcal{D}$ if, for each solution $\sigma \in Sol(\mathcal{O})$ and each piecewise-interchangeable bijection b over $\mathcal{D}$, $b \circ \sigma \in Sol(\mathcal{O})$ and $f(\sigma) = f(b \circ \sigma)$.

In the following, we often assume fixed sets V and D in examples for simplicity and talk directly about the composition and interchangeability of constraints, since they are essentially equivalent to their CSP counterparts, as we observe that if $\mathcal{P}_1 = \langle V, D, C_1\rangle$ and $\mathcal{P}_2 = \langle V, D, C_2\rangle$ are two CSPs, then their composition $\mathcal{P}_1 \wedge \mathcal{P}_2 = \langle V, D, C_1 \wedge C_2\rangle$. Using the definitions of value interchangeability, we will see how symmetries for a composition of constraints can be inferred from the individual symmetries of each constraint.

Proposition 13 (Proposition 1) Let $\mathcal{P}_1 = \langle V, D, C_1\rangle$ and $\mathcal{P}_2 = \langle V, D, C_2\rangle$ be two value interchangeable CSPs. Then, their composition $\mathcal{P}_1 \wedge \mathcal{P}_2$ is value interchangeable.

Example 14 (Example 4) Let $V \supseteq \{v_1, \ldots, v_6\}$ and let C_1 and C_2 be the constraints allDifferent(v_1, v_2, v_3) and allDifferent(v_4, v_5, v_6). Then $C_1 \wedge C_2$ is value interchangeable.

As in the previous example, it is often the case that a constraint does not constrain all variables, but while this does not affect the value interchangeability, it might affect the variable interchangeability of the problem as we will see in Section 7.

Proposition 15 (Proposition 2) Let $\mathcal{P}_1 = \langle V, D, C_1 \rangle$ and $\mathcal{P}_2 = \langle V, D, C_2 \rangle$ be two CSPs. Assume that $\mathcal{P}_i$ is piecewise value interchangeable over partition $\mathcal{D}_i$ of D $(1 \leq i \leq 2)$. Then the composition $\mathcal{P}_1 \wedge \mathcal{P}_2$ is piecewise value interchangeable over

$$\mathcal{D} = \{D_1 \cap D_2 \mid D_1 \in \mathcal{D}_1 \wedge D_2 \in \mathcal{D}_2 \wedge D_1 \cap D_2 \neq \emptyset\}.$$

Proof. First observe that $\mathcal{D}$ is a partition of D. Now let b be a piecewise-interchangeable bijection over $\mathcal{D}$. We show that b is piecewise-interchangeable over $\mathcal{D}_1$. Indeed, consider a set $D_1 \in \mathcal{D}_1$ and a value $d \in D_1$. By definition of $\mathcal{D}$, there exists $D_2 \in \mathcal{D}_2$ such that $I = D_1 \cap D_2$ and $d \in I$. Since b is piecewise-interchangeable over $\mathcal{D}$, $b(d) \in I \subseteq D_1$ and b is piecewise-interchangeable over $\mathcal{D}_1$. Similarly, we can show that b is piecewise-interchangeable over $\mathcal{D}_2$. As a consequence, if $\sigma \in Sol(\mathcal{P}_1 \wedge \mathcal{P}_2)$, then $b \circ \sigma \in Sol(\mathcal{P}_1)$ and $b \circ \sigma \in Sol(\mathcal{P}_2)$. Hence, $b \circ \sigma \in Sol(\mathcal{P}_1 \wedge \mathcal{P}_2)$.

Example 16 (Example 5) Let $D = \{1, \ldots, 10\}$ and let C_1 and C_2 be the constraints atmost$(1, 1, \langle v_1, \ldots, v_5 \rangle)$ and atmost$(2, 2, \langle v_1, \ldots, v_5 \rangle)$ which are PVI over $\mathcal{D}_1 = \{\{1\}, \{2, \ldots, 10\}\}$ and $\mathcal{D}_2 = \{\{2\}, \{1, 3, \ldots, 10\}\}$ respectively. The composition $C_1 \wedge C_2$ is PVI over

$$\mathcal{D} = \{\{1\}, \{2\}, \{3, \ldots, 10\}\}.$$

Analogously to the composition of constraint symmetries, we will now see how symmetries can be derived for composition for functions, COPs and numerical constraints.

Proposition 17 (Proposition 3) Let f_1 and f_2 be two global functions of signature $(V \to D) \to \mathbb{N}$. If f_1 and f_2 are value interchangeable, then so is $f_1 \star f_2$ $(\star \in \{+, -, \times\})$.

Example 18 (Example 6) Let $V \supseteq \{v_1, \ldots, v_6\}$ and let f_1 and f_2 be the global functions nbDistinct(v_1, v_2, v_3) and nbDistinct(v_4, v_5, v_6). Then, the global function $3f_1 + 4f_2$ is value interchangeable.

Proposition 19 (Proposition 4) Let $f_1 : (V \to D) \to \mathbb{N}$ and $f_2 : (V \to D) \to \mathbb{N}$ be two global functions. If f_1 and f_2 are piecewise value interchangeable over $\mathcal{D}_1$ and $\mathcal{D}_2$ respectively, then $f_1 \star f_2$, where $\star \in \{+, -, \times\}$, is piecewise value interchangeable over

$$\mathcal{D} = \{D_1 \cap D_2 \mid D_1 \in \mathcal{D}_1 \wedge D_2 \in \mathcal{D}_2 \wedge D_1 \cap D_2 \neq \emptyset\}.$$

Proposition 20 (Proposition 5) Let $\mathcal{O} = \langle V, D, C, f\rangle$ be a *COP* and $\mathcal{P} = \langle V, D, C\rangle$. If $\mathcal{P}$ and f are value interchangeable, then $\mathcal{O}$ is value interchangeable. If $\mathcal{P}$ is piecewise value interchangeable over partition $\mathcal{D}_1$ of D and f is piecewise value interchangeable over partition $\mathcal{D}_2$ of D, then $\mathcal{O}$ is piecewise value interchangeable over

$$\mathcal{D} = \{D_1 \cap D_2 \mid D_1 \in \mathcal{D}_1 \wedge D_2 \in \mathcal{D}_2 \wedge D_1 \cap D_2 \neq \emptyset\}.$$

Proposition 21 (Proposition 6) Let $f : (V \to D) \to \mathbb{N}$ be a global function and $\mathcal{D}$ be a partition of D. If f is piecewise value interchangeable over $\mathcal{D}$, then the CSP $\langle V, D, f \approx 0\rangle$ is piecewise value interchangeable over $\mathcal{D}$ as well, where $\approx \in \{>, \geq, =, \neq, \leq, <\}$.

From this proposition and Proposition 19 we can derive the following proposition, which is not in [4]:

Proposition 22 If $f : (V \to D) \to \mathbb{N}$ and $g : (V \to D) \to \mathbb{N}$ are two global functions that are PVI over the partitions $\mathcal{D}_1$ and $\mathcal{D}_2$ of D, then the CSP $\langle V, D, f \approx g\rangle$, where $\approx \in \{>, \geq, =, \neq, \leq, <\}$, is PVI over $\mathcal{D} = \{D_1 \cap D_2 \mid D_1 \in \mathcal{D}_1 \wedge D_2 \in \mathcal{D}_2 \wedge D_1 \cap D_2 \neq \emptyset\}$.

5 Value Interchangeability in Single Constraints

In this section, we will derive (piecewise) value interchangeability, as defined in the previous section, for the problems of Section 3. If we look at the different relational models in Sections 3.1 to 3.6, we can easily see that there are certain types of constraints that occur more than once, and we shall look at these to detect common patterns and derive symmetries for them in order to be able to detect symmetries for similar constraints in other problems.

5.1 Value Interchangeable Count Constraints, Example 1

The first constraint in the relational model of the progressive party problem (Figure 2)

$$\forall(g : \mathit{Guests}, h : \mathit{Hosts})\ \mathrm{count}(0 \ldots 1)(p : \mathit{Periods} \mid \mathit{schedule}(g, p) = h)$$

and the constraint in the scene allocation problem (Figure 4)

$$\forall(d : \mathit{Days})\ \mathrm{count}(0 \ldots 5)(s : \mathit{Scenes} \mid \mathit{shoot}(s) = d)$$

look very similar. In both of the problems the following pattern occurs:

$$\begin{array}{l} \mathrm{var}\ \ldots \longrightarrow \mathit{Domain} \\ \vdots \\ \forall(\ldots d : \mathit{Domain} \ldots)\ \mathrm{count}(0 \ldots k)(\ldots \mid \ldots = d) \end{array}$$

Since the constraint holds for all d in the domain, we can conclude that if σ is a solution to a CSP with a constraint of this pattern and $b : D \longrightarrow D$ is a bijection on the domain of the problem, then $b \circ \sigma$ is also a solution, i.e. this constraint is fully value interchangeable. But, of course, it is important that count(...) holds for all the elements in the domain.

Further, we also see that the value of the constant k does not affect the symmetry of the constraint. In fact, it does not even matter if there are at least, exactly, or at most k variables such that the constraint holds. In fact not only sets of the type $\{k, \ldots, l\}$ are allowed in the count expression, but all kinds of sets, and it is straightforward to see that in the pattern

$$\begin{aligned}&\text{var } \ldots \longrightarrow \mathit{Domain}\\ &\vdots\\ &\forall(\ldots d : \mathit{Domain} \ldots)\text{count}(\langle \mathit{Set}\rangle)(\ldots \mid \ \ldots = d)\end{aligned}$$

the set does not affect the symmetry of the constraint at all, as long as it does not contain any explicit references to d, since it is not in any way dependent on the elements of the domain.

Thus we can conclude that relational constraints with the following pattern are fully value interchangeable (the syntax used is found in [2]):

$$\begin{aligned}&\texttt{var}\langle \mathit{Id}\rangle : \langle \mathit{SetExpr}\rangle \ \texttt{-> Domain}\\ &\vdots\\ &\texttt{forall(}\langle \mathit{QuantExpr}\rangle\texttt{)count(}\langle \mathit{Set}\rangle\texttt{)}\\ &\texttt{((}(\langle \mathit{RelQvars}\rangle \mid \langle \mathit{IdTuple}\rangle^{\texttt{\&}^+}\texttt{)in}\langle \mathit{SetExpr}\rangle)^{\texttt{,}^+} \mid \langle \mathit{NumExpr}\rangle \ \texttt{= d)}\end{aligned} \tag{7}$$

where

1. $\langle \mathit{QuantExpr}\rangle$ contains an expression `d:Domain`, and where this expression is preceeded by a comma or a parenthesis and not followed by |. This is to avoid expressions such as: $\forall(3 < d : \mathit{Domain})$ or $\forall(d : \mathit{Domain} \mid d < 3)$ which would break the full symmetry.
2. $\langle \mathit{Set}\rangle$ does not contain any explicit references to d.

Now we can use this pattern to check for symmetries in other problems, and we note that the second constraint in the social golfers problem (Figure 5) is of this type and we can thus conclude that it is fully value-symmetric.

If we on the other hand look at the warehouse problem (Figure 3.6)

$$\begin{aligned}&\text{var } \mathit{supplier} : \mathit{Stores} \longrightarrow \mathit{Warehouses}\\ &\vdots\\ &\forall(w : \mathit{Warehouses})\ \text{count}(0 \ldots \mathit{cap}(w))(s : \mathit{Stores} \mid \mathit{supplier}(s) = w)\end{aligned}$$

we see that the constraint in this model is very similar to our pattern (7), but we also see that restriction 2 above is violated. In this problem, the values of the

domain – the warehouses – are not indistinguishable, but only warehouses with the same capacity are interchangeable. The warehouse problem is, modelled this way, thus *piecewise* value interchangeable over partitions of the warehouses with the same capacity. To detect this, and other PVI constraints similar to this one, we need a more general form of pattern (7). A more general constraint than constraint (7), with its restrictions, would be:

$$\begin{array}{l} \text{var } \ldots \longrightarrow \mathit{Domain} \\ \vdots \\ \underbrace{\forall(\ldots d : D \ldots)}_{1}\ \underbrace{\text{count}(\langle \mathit{Set} \rangle)}_{2}\ \underbrace{(\ldots \mid\ \ldots = f(d))}_{3} \end{array} \tag{8}$$

where $D \subseteq \mathit{Domain}$, $\langle \mathit{Set} \rangle$ is some set, possibly expressed with a reference to d, and f is some function in $\mathit{Domain} \longrightarrow \mathbb{N}$ (the range of the function could of course also be the integers or the reals). We can divide the constraint into three parts, where each one of them, depending on their formulation, can break the full symmetry. Let us assume that we have full symmetry for the constraint in order to see how this symmetry is affected as *one* of these three parts are changed in a way that breaks the symmetry. In the cases 1–3 below, we will say that a part of the constraint is "value interchangeable" or "PVI" over some partition $\mathcal{D}$, meaning that, the rest of the constraint being equal to constraint (7), this or that formulation of the considered part makes the whole constraint value interchangeable or PVI over $\mathcal{D}$.

1. The first part of the constraint is the $\forall(\ldots d : D \ldots)$ expression which is "PVI" over $\mathcal{D} = \{D, \mathit{Domain} \setminus D\}$.

   Proof. Assume that we have a CSP, $\mathcal{P}$ with a constraint like (8), where the second and third parts of the constraint are fully symmetric, and that σ is a solution to $\mathcal{P}$. Now, for any bijection b such that if $d \in D$, then $b(d) \in D$, $b \circ \sigma$ will also be a solution, since the constraint holds for all $d \in D$. That is, $\mathcal{P}$ is PVI over any partition $\mathcal{D}$ of *Domain* such that for any piecewise interchangeable bijection, b, over $\mathcal{D}$, if $d \in D$, then $b(d) \in D$, and indeed $\mathcal{D} = \{D, \mathit{Domain} \setminus D\}$ is such a partition, and in fact the coarsest one. With our terminology, this means that $\forall(\ldots d : D \ldots)$ is "PVI" over $\mathcal{D}$.

   In the case where $D = \mathit{Domain}$ it is "PVI" over $\mathcal{D} = \{\mathit{Domain}\}$ and we have full symmetry, as established above. To express $\forall(\ldots d : D \ldots)$, where $D \subseteq \mathit{Domain}$, we have several possibilities:

   (a) $\forall(k > d : \mathit{Domain}) \equiv \forall(d : \mathit{Domain} \mid d < k)$, which is "PVI" over $\mathcal{D} = \{\{d : \mathit{Domain} \mid d < k\}, \{d : \mathit{Domain} \mid d \geq k\}\}$.

   (b) $\forall(k \geq d : \mathit{Domain}) \equiv \forall(d : \mathit{Domain} \mid d \leq k)$, which is "PVI" over $\mathcal{D} = \{\{d : \mathit{Domain} \mid d \leq k\}, \{d : \mathit{Domain} \mid d > k\}\}$.

   (c) $\forall(k = d : \mathit{Domain}) \equiv \forall(d : \mathit{Domain} \mid d = k)$, which is "PVI" over $\mathcal{D} = \{\{k\}, \{\mathit{Domain} \setminus \{k\}\}\}$.

(d) $\forall(k \neq d : \mathit{Domain}) \equiv \forall(d : \mathit{Domain} \mid d \neq k)$, which is "PVI" over $\mathcal{D} = \{\{k\}, \{\mathit{Domain} \backslash \{k\}\}\}$.

(e) $\forall(k \leq d : \mathit{Domain}) \equiv \forall(d : \mathit{Domain} \mid d \geq k)$, which is "PVI" over $\mathcal{D} = \{\{d : \mathit{Domain} \mid d \geq k\}, \{d : \mathit{Domain} \mid d < k\}\}$.

(f) $\forall(k < d : \mathit{Domain}) \equiv \forall(d : \mathit{Domain} \mid d > k)$, which is "PVI" over $\mathcal{D} = \{\{d : \mathit{Domain} \mid d > k\}, \{d : \mathit{Domain} \mid d \leq k\}\}$.

Note that (1a) and (1e) are "PVI" over the same partition of the values, as are (1b) and (1f), and (1c) and (1d) respectively. If we think about it, the expressions $\forall(d : \mathit{Domain} \mid d \approx k)$ can be generalised even more, viz. if we have

(g) $\forall(d : \mathit{Domain} \mid \langle \mathit{Formula} \rangle)$ where $\langle \mathit{Formula} \rangle$ is some formula, possibly refering to d. This would be "PVI" over the partition $\mathcal{D} = \{D, \mathit{Domain} \setminus D\}$, where $D = \{d : \mathit{Domain} \mid \langle \mathit{Formula} \rangle\}$.

It is also possible to combine any of (1a)–(1f) with (1g):

(h) $\forall(k \approx d : \mathit{Domain} \mid \langle \mathit{Formula} \rangle)$, where $\approx \in \{<, \leq, =, \neq, \geq, >\}$. This expression is in the general case "PVI" over $\mathcal{D} = \{D_1 \cap D_2 \mid D_1 \in \mathcal{D}_1 \ \wedge \ D_2 \in \mathcal{D}_2 \ \wedge \ D_1 \cap D_2 \neq \emptyset\}$ if $\forall(k \approx d : \mathit{Domain})$ is "PVI" over $\mathcal{D}_1$ and $\forall(d : \mathit{Domain} \mid \langle \mathit{Formula} \rangle)$ is "PVI" over $\mathcal{D}_2$.

Example. The expression $\forall(2 < d : \mathit{Domain} \mid f(d) \geq 20)$, where $\mathit{Domain} = \{1, 2, 3, 4\}$ and $f(1) = f(3) = f(4) = 25$, $f(2) = 1$, is PVI over $\mathcal{D} = \{\{1\}, \{2\}, \{3, 4\}\}$.

But if we think about it further, this expression only considers the elements $d : \mathit{Domain}$ s.t. $d > 2$ and $f(d) \geq 20$; for all the elements of Domain for which any of these conditions does not hold, we do not have to make any further partition. That is, in our example, we can make a coarser partition of the elements of the domain and conclude that the expression is "PVI" over $\mathcal{D}' = \{\{1, 2\}, \{3, 4\}\}$. By thus exploiting the semantics of the forall-expression, we can in many cases find a better partition than we would using only the general result in (1h) above. In cases (1i) and (1j) below we list some such cases where the expression is "PVI" over a coarser partition than in the general case (there are many such cases, depending on what $\langle \mathit{Formula} \rangle$ looks like, it could for example be a conjunction of formulas including d, a case which is not discussed below):

(i) $\forall(k_1 \approx_1 d : \mathit{Domain} \mid f(d) \approx_2 k_2)$, where $\approx_1, \approx_2 \in \{<, \leq, =, \neq, \geq, >\}$ is "PVI" over $\mathcal{D} = \{D, \mathit{Domain} \setminus D\}$, where $D = \{d : \mathit{Domain} \mid k_1 \approx_1 d \wedge f(d) \approx_2 k_2\}$.

(j) $\forall(k_1 \approx_1 d : \mathit{Domain} \mid k_2 \approx_2 f(d))$, where $\approx_1, \approx_2 \in \{<, \leq, =, \neq, \geq, >\}$ is "PVI" over $\mathcal{D} = \{D, \mathit{Domain} \setminus D\}$, where $D = \{d : \mathit{Domain} \mid k_1 \approx_1 d \wedge k_2 \approx_2 f(d)\}$.

2. Our next task is to decide in what way the $\langle Set \rangle$ in the count expression affects the symmetry of the constraint. In Pattern 1 we draw the conclusion that as long as there are no explicit references to d in the set expression, it does not affect the symmetry of the constraint. There are many possible ways to declare different sets, but at least the following are of interest:

   (a) count$(k \dots d)$, where $k \in \mathbb{N}$

   (b) count$(d \dots k)$

   (c) count(d)

   The reference to d in the expressions (2a)–(2c) completely breaks the symmetry and makes the expression "PVI" over the partition $\mathcal{D} = \{\{d_i\} \mid d_i : \mathit{Domain}\}$.

   (e) count$(k \dots f(d))$, where $k \in \mathbb{N}$ and f is some function in $\mathit{Domain} \longrightarrow \mathbb{N}$

   (f) count$(f(d) \dots k)$

   (g) count$(f(d))$

   The expressions in (2e)-(2g) are "PVI" over $\mathcal{D} = \{\{d : \mathit{Domain} \mid f(d) = f(d_i)\} \mid d_i : \mathit{Domain}\}$, i.e. over the partition where elements of the domain with the same value for f are grouped together.

3. Finally, we look at the last part of the constraint, i.e. $(\dots \mid \dots = f(d))$. We know that f is some function in $\mathit{Domain} \longrightarrow \mathbb{N}$, and suppose that $D = \{f(d) \mid d : \mathit{Domain}\} \cap \mathit{Domain}$. Then the third part of the constraint is "PVI" over $\mathcal{D} = \{D, \mathit{Domain} \setminus D\}$. In particular, if $f = id$, i.e. if the whole expression is $(\dots \mid \dots = d)$, then $D = \mathit{Domain}$ and $\mathcal{D} = \{\mathit{Domain}\}$, i.e. it is fully "value interchangeable", as established above.

Using the result in (2e) above, we can conclude that the constraint in the model of the warehouse location problem in Figure 3.6) is PVI over $\{\{w : \mathit{Warehouses} \mid \mathit{cap}(w) = \mathit{cap}(w_i)\} \mid w_i : \mathit{Warehouses}\}$, i.e. over the partition of warehouses with the same capacity.

But how is the symmetry of a constraint of pattern (7) affected if more than one of the three parts of the constraint differs from (7)?

Proposition 23 Let us suppose that we have a constraint, such that the first part is "PVI" over $\mathcal{D}_1$, the second is "PVI" over $\mathcal{D}_2$, and the third is "PVI" over $\mathcal{D}_3$. Then the whole constraint is PVI over $\mathcal{D} = \{D_1 \cap D_2 \cap D_3 \mid D_1 \in \mathcal{D}_1 \ \wedge \ D_2 \in \mathcal{D}_2 \ \wedge \ D_3 \in \mathcal{D}_3 \ \wedge \ D_1 \cap D_2 \cap D_3 \neq \emptyset\}$.

We can now summarise the results of this section into our first general pattern that can be used to detect value interchangeability:

Pattern 1:
var $\ldots \longrightarrow \mathit{Domain}$
$\vdots$
$\forall(\ldots d : D \ldots)\text{count}(\langle \mathit{Set} \rangle)(\ldots \mid \ \ldots = f(d))$

Constraints of this pattern are PVI over $\mathcal{D} = \{D_1 \cap D_2 \cap D_3 \mid D_1 \in \mathcal{D}_1 \ \wedge \ D_2 \in \mathcal{D}_2 \ \wedge \ D_3 \in \mathcal{D}_3 \ \wedge \ D_1 \cap D_2 \cap D_3 \neq \emptyset\}$ if $\mathcal{D}_1 = \{D, \mathit{Domain} \setminus D\}$, $\mathcal{D}_2 = \mathit{Domain}$ if we have no references to d in the set $\langle \mathit{Set} \rangle$ or if $\langle \mathit{Set} \rangle = \{k_1, k_2, \ldots, k_m, f(d), k_{m+1}, \ldots k_n\}$ where $k_1 \ldots k_n \in \mathbb{N}$, then $\mathcal{D}_2 = \{\{d : \mathit{Domain} \mid f(d) = f(d_i)\} \mid d_i : \mathit{Domain}\}$, and $\mathcal{D}_3 = \{S, \mathit{Domain} \setminus S\}$, where $S = \{f(d) \mid d : \mathit{Domain}, f(d) : \mathit{Domain}\}$.

Example 24 Suppose that we have a model of a CSP $\mathcal{P} = \langle V, D, C \rangle$ which includes the following constraint:

$$\underbrace{\forall(2 < d : D \mid d \neq 6)}_{1}\underbrace{\text{count}(0 \ldots f(d))}_{2}\underbrace{(v : V \mid f(v) = G(d))}_{3} \tag{9}$$

Let us further suppose that $D = \{1, 2, \ldots, 10\}$ and f and G are two functions defined on D, s.t. $f(d) = 0$ if $d \leq 5$ and $f(d) = 1$ otherwise and $G(d) = d{+}2$, $d \in D$. Then, by (1i) above, part 1 is "PVI" over $\mathcal{D}_1 = \{\{1, 2, 6\}, \{3, 4, 5, 7, 8, 9, 10\}\}$, by (2e) above, part 2 is "PVI" over $\mathcal{D}_2 = \{\{1, 2, \ldots, 5\}, \{6, 7, \ldots, 10\}\}$ and by (3) above is part 3 "PVI" over $\mathcal{D}_3 = \{\{1, 2, \ldots, 8\}, \{9, 10\}\}$. From this and Proposition 23 we derive that the whole constraint 9 is PVI over $\mathcal{D} = \{\{1, 2\}, \{3, 4, 5\}, \{6\}, \{7, 8\}, \{9, 10\}\}$.

5.2 Value Interchangeable Count Constraints, Example 2

Let us now look at the second constraint of the progressive party problem in Figure 2:

var $\mathit{schedule} : (\mathit{Guests} \times \mathit{Periods}) \longrightarrow \mathit{Hosts}$
$\vdots$
$\forall(i < j : \mathit{Guests})\ \text{count}(0 \ldots 1)(p : \mathit{Periods} \mid \mathit{schedule}(i, p) = \mathit{schedule}(j, p))$

Whereas the first constraint in the model of the progressive party problem (Figure 2) had similarities to the constraint of the scene allocation problem (Figure 4), the second constraint, on its hand, is similar to the first constraint of the model of the social golfers problem (Figure 5):

var $\mathit{schedule} : (\mathit{Players} \times \mathit{Weeks}) \longrightarrow^{s*w} \mathit{Groups}$
$\vdots$
$\forall(p < q : \mathit{Players})\ \text{count}\ (0 \ldots 1)(v : \mathit{Weeks} \mid \mathit{schedule}(p, v) = \mathit{schedule}(q, v))$

In these two examples, we note that there is no explicit reference to any elements of the domain in the constraints and this indicates that they are value-symmetric and that the forall expression and the set in the count expression are not part of the symmetric pattern.

Proof. Let us assume that we have a CSP $\mathcal{P} = \langle V = \{v_1, v_2, \ldots, v_n\}, D, C\rangle$. Now let

$$\text{var } f : V \longrightarrow D$$

be a variable declaration in a relational model of the problem. Let us then construct a partition of V, $\mathcal{V} = \{V_1, V_2, \ldots, V_n\}$, where $V_i = \{v : V \mid f(v) = f(v_i)\}$. Now suppose we have a bijection $b : D \longrightarrow D$, then if $f(v) = f(v_i)$ for $v, v_i \in V$ then $(b \circ f)(v) = (b \circ f)(v_i)$, which means that $V_i = V_{i_b}$, where $V_{i_b} = \{v : V \mid (b \circ f)(v) = (b \circ f)(v_i)\}$, i.e. the partition $\mathcal{V}$ is unchanged. So if we take any subsets, W_1 and W_2, of V, then for all $w_1 \in W_1$ we have that $|\{w_2 : W_2 \mid f(w_1) = f(w_2)\}|$ is not affected by the bijection b.

This shows not only that the problem is value interchangeable, but also that the only thing that defines the pattern in these constraints is the function f, and the count expression. Using the syntax from [2], we can conclude that constraints of the following pattern are fully value-symmetric:

$$\begin{array}{l}
\texttt{var f : } \langle \textit{SetExpr1} \rangle \\
\vdots \\
[\texttt{forall(}\langle \textit{QuantExpr} \rangle\texttt{)}]\texttt{count(}\langle \textit{Set} \rangle\texttt{)} \\
\texttt{(((}\langle \textit{RelQvars} \rangle\texttt{|}\langle \textit{IdTuple} \rangle^{\texttt{\&}^+}\texttt{)in}\langle \textit{SetExpr2} \rangle\texttt{)}^{\texttt{,}^+} \texttt{ | f(}\langle \textit{Tuple} \rangle\texttt{) = f(}\langle \textit{Tuple} \rangle\texttt{))}
\end{array} \tag{10}$$

But as in the constraint (7) of Section 5.1, there are some restrictions if we want to have *full* value interchangeability. There can be no explicit references to f in any of the expressions $\langle \textit{QuantExpr} \rangle$, $\langle \textit{Set} \rangle$, $\langle \textit{SetExpr1} \rangle$ or $\langle \textit{SetExpr2} \rangle$ in the count constraint. Otherwise we could have a CSP:

$$\begin{array}{l}
\text{var } f : V \longrightarrow D \\
\vdots \\
\forall(v : V \mid f(v) < k)\ \text{count}(0 \ldots 1)(w : V \mid f(v) = f(w))
\end{array}$$

which is not fully value interchangeable as we will see below.

Can we use this pattern to say something about the symmetry of the graph colouring problem? Since it is not the actual colours of the nodes in the graph, but rather which nodes have the same colours, that matters, the problem is fully value interchangeable, and the constraint in the model in Figure 6 in fact does look something like the pattern (10), but not exactly. It rather has the pattern:

$$\begin{array}{l}
\text{var } f : V \longrightarrow D \\
\vdots \\
\forall(\ldots v : V \ldots)\ \text{count}(\ldots)(w : V \mid \langle \textit{Expr} \rangle \wedge f(v) = f(w))
\end{array}$$

If the expression $\langle Expr \rangle$ does not contain any references to f, it does not affect the value interchangeability of the constraint, which is thus the case in the model in Figure 6.

Now, let us, just as in Section 5.1, see in what way the symmetry of a constraint of pattern (10) is affected if one of the restrictions above is violated, i.e. when a reference to f in any of the expressions mentioned above breaks the full symmetry into a partial one. Again the constraint can be divided into three parts, and we will investigate how the symmetry of the constraint is affected if one of these parts is changed compared with constraint (10) and the others remain the same, using "value interchangeable" and "PVI" in the same meaning as in Section 5.1. A generalisation of constraint (10) could be:

$$\begin{array}{l}\text{var } f : V \longrightarrow Domain \\ \vdots \\ \underbrace{\forall(v : V \mid \ldots f(v) \ldots)}_{1}\ \underbrace{\text{count}(\ldots f(v) \ldots)}_{2}\underbrace{((w : V \mid \ldots f(v) \ldots) \mid f(v) = f(w))}_{3}\end{array}$$

Examining each of the three parts separately, we have the three following cases:

1. We have already concluded that we have full value interchangeability if there is no restriction to the set of variables that we are looking at, i.e. if part 1 looks like $\forall(v : V)$. If this is not the case, we have at least two possibilities:

   (a) $\forall(v : V \mid f(v) \approx k)$, where $k \in \mathbb{N}$ and $\approx \in \{<, \leq, =, \neq, \geq, >\}$. This is "PVI" over $\mathcal{D} = \{D, Domain \setminus D\}$, where $D = \{d : Domain \mid d \approx k\}$.

   (b) $\forall(v : V \mid k \approx f(v))$. If $D = \{d : Domain \mid k \approx d\}$, this is "PVI" over $\mathcal{D} = \{D, Domain \setminus D\}$.

2. Without references to $f(v)$ in the set of the count expression, we have full value interchangeability. Other than this, there are at least three plausible cases:

   (a) $\text{count}(k \ldots f(v))$, where $k \in \mathbb{N}$

   (b) $\text{count}(f(v) \ldots k)$

   (c) $\text{count}(f(v))$

   These three cases are all "PVI" over the trivial partition of the domain, i.e. $\mathcal{D} = \{\{d_i\} \mid d_i : Domain\}$.

3. In the third part of the constraint, we have, just as in the first part, a possibility to restrict the set of variables which the constraint concerns. If this is not done, the constraint is fully symmetric. Otherwise we have two cases analogous to the ones in (1):

(a) $((w : V \mid f(v) \approx k) \mid f(v) = f(w))$, where $k \in \mathbb{N}$ and $\approx \in \{<, \leq, =, \neq, \geq, >\}$. This is "PVI" over $\mathcal{D} = \{D, \mathit{Domain} \setminus D\}$, where $D = \{d : \mathit{Domain} \mid d \approx k\}$.

(b) $((w : V \mid k \approx f(v)) \mid f(v) = f(w))$, which, if $D = \{d : \mathit{Domain} \mid k \approx d\}$ is "PVI" over $\mathcal{D} = \{D, \mathit{Domain} \setminus D\}$.

To conclude this section, we summarise this into a general pattern, using Proposition 23 in Section 5.1.

Pattern 2:
var $f : V \longrightarrow \mathit{Domain}$
$\vdots$
$\forall(v : V \mid \ldots f(v) \ldots)\text{count}(\ldots f(v) \ldots)((w : V \mid \ldots f(v) \ldots) \mid f(v) = f(w))$

Constraints of this pattern are PVI over $\mathcal{D} = \{D_1 \cap D_2 \cap D_3 \mid D_1 \in \mathcal{D}_1 \ \wedge \ D_2 \in \mathcal{D}_2 \ \wedge \ D_3 \in \mathcal{D}_3 \ \wedge \ D_1 \cap D_2 \cap D_3 \neq \emptyset\}$ where $\mathcal{D}_1$ corresponds to $\mathcal{D}$ in case 1 above, $\mathcal{D}_2$ corresponds to $\mathcal{D}$ in case 2 above, and $\mathcal{D}_3$, finally, corresponds to $\mathcal{D}$ in case 3 above.

5.3 Symmetries in Objective Functions and Numerical Constraints

Using the results of Sections 5.1 and 5.2 we have been able to detect value symmetries for all constraints in our models (except, of course, in the n-Queens and the BIBD problem, which are only value interchangeable over the trivial partition of the values) with one exception, namely for the numerical constraints and for objective functions occuring in COPs. In order to compositionally derive the symmetries for whole problems, we have to be able to detect symmetries in objective functions. As in Sections 5.1 and 5.2 we will first look at some examples and then, with the Definitions 8 and 10 and Propositions 17 and 19, see what symmetries they exhibit. Our first example will be the third constraint of the progressive party problem in Figure 2:

$$\underbrace{\forall(p : \mathit{Periods},\ h : \mathit{Hosts})}_{1} \underbrace{\left(\sum_{g:\mathit{Guests} \ \mid \ \mathit{schedule}(g,p)=h} \mathit{crewSize}(g) \right)}_{2} \leq \underbrace{\mathit{spareCap}(h)}_{3} \quad (11)$$

As indicated, we will use the same approach as in Sections 5.1 and 5.2 and divide the constraint (11) into three parts and study their symmetries separately first and then derive the symmetries for the whole constraint. In the first part we see that no restrictions are made to the set of *Hosts* from which the element h can be chosen, and it is thus fully "value interchangeable". Now, if we look at the rest of the constraint, we see that it is a numerical constraint, which by Proposition 22 is PVI over the partition $\mathcal{D} = \{D_1 \cap D_2 \mid D_1 \in \mathcal{D}_1 \wedge D_2 \in \mathcal{D}_2 \wedge D_1 \cap D_2 \neq \emptyset\}$,

if $\left(\sum_{g:Guests \mid schedule(g,p)=h} crewSize(g)\right)$ is PVI over $\mathcal{D}_1$ and $spareCap(h)$ is PVI over $\mathcal{D}_2$. What can we say about the symmetries of these expressions? Since in the summation function, there is no reference to any specific hosts (the host h in the summation range is not specified here, but in the forall expression that preceeds the summation expression) in the summands or in the summation range, any bijection on the hosts will not affect the value of the function, and it is thus fully "value interchangeable" according to Definition 8. We can generalise this in a pattern that can be applied to other functions as well:

Pattern 3: A function with any of the following patterns: $\sum_{\langle QuantExpr\rangle} \langle NumExpr\rangle$, abs$\langle NumExpr\rangle$, card$\langle SetExpr\rangle$, or $f\langle Tuple\rangle$, where the expressions $\langle QuantExpr\rangle$, $\langle NumExpr\rangle$, $\langle SetExpr\rangle$ or $\langle Tuple\rangle$ do not contain any references to any specific elements of the domain, is fully "value interchangeable".

In the third part of constraint (11) we have a function *spareCap* with the signature $Hosts \longrightarrow \mathbb{N}$. This means that $spareCap(h)$ is PVI over the partition $\mathcal{D} = \{\{h : Hosts \mid spareCap(h) = spareCap(h_i)\} \mid h_i \in Hosts\}$. This can also be turned into a general pattern:

Pattern 4: If the following occurs in a relational model of a CSP:

$$\begin{aligned} &\text{cst } f : Domain \longrightarrow \mathbb{N} \\ &\vdots \\ &\ldots d : Domain \ldots f(d) \end{aligned}$$

then the function $f(d)$ is "PVI" over the partition $\mathcal{D} = \{\{d : Domain \mid f(d) = f(d_i)\} \mid d_i \in Domain\}$.

From this, we can finally, using Propositions 22 and 23, draw the conclusion that the constraint (11) is PVI over the partition $\mathcal{D} = \{\{h : Hosts \mid spareCap(h) = spareCap(h_i)\} \mid h_i \in Hosts\}$.

Our next example is not a numerical constraint, but the objective function occurring in the scene allocation problem. The function has the following form:

$$\sum_{a:Actors} fee(a) * \text{card}\{(d : Days) \mid \exists (s : Scenes \mid casting(a, s) \wedge shoot(s) = d)\} \tag{12}$$

The first part of the summands in the function, i.e. $fee(a)$, does not affect the symmetry of the function since we do not have any reference to any elements from *Days* there. Then we have a cardinality expression with references to elements from the domain. The way the scene allocation problem is stated, it is easy to see that the actual days assigned to the scenes are not important, but as we want a pattern for more general cardinality expressions, we will study them more closely.

What does it mean for a function of the form $f(d) = \text{card}\langle SetExpr\rangle$ to be PVI over some partition $\mathcal{D}$? Well, suppose that $f(d) = n$, then $f(d)$ is PVI over

$\mathcal{D}$ if for any piecewise interchangeable bijection $b : \mathcal{D} \longrightarrow \mathcal{D}$, $f(d) = (f \circ b)(d)$, i.e. if $f(b(d)) = n$. But to say that $\text{card}\langle \mathit{SetExpr} \rangle = n$ can be expressed by saying that $\text{count}(n)(x \mid x \in \langle \mathit{SetExpr} \rangle)$, and these two expressions are thus PVI over the same partition. (Note that n is an arbitrary number that does not affect the symmetry of the expressions.) For our constraint (12), for example, the cardinality expression is PVI over the same partition as the constraint:

$$\text{count}(n)((d : \mathit{Days}) \mid \exists(s : \mathit{Scenes} \mid \mathit{casting}(a, s) \wedge \mathit{shoot}(s) = d)) \tag{13}$$

We have thus transformed the question of symmetry for the cardinality expression into a question on symmetry of a count constraint. Those we have studied before, but only when preceeded by a universal quantifier. Therefore we will look at the symmetry for some "simple" count constraints. If we look at the objective function of the model of the warehouse location problem, we find a cardinality expression very similar to the one in the scene allocation problem. In what way can we generalise their "corresponding" count constraints? The following pattern is at least one generalisation of constraint (13):

Pattern 5:
var $f : \mathit{Vars} \longrightarrow \mathit{Domain}$
$\vdots$
$\text{count}(S)((d : D) \mid \exists(v : \mathit{Vars} \mid \mathit{Form} \wedge f(v) = d))$

where $S \subseteq \mathbb{N}$, $D \subseteq \mathit{Domain}$, and Form is some formula without references to elements from the domain. Now, the only part of this constraint that can affect its value interchangeability is the expression $d : D$. We can, as in Section 5.1, conclude that this part, and thus the whole constraint, is PVI over the partition $\mathcal{D} = \{D, \mathit{Domain} \setminus D\}$, where we can apply the cases (1a)-(1j) of Section 5.1 if we want to explicitly express some possibilities for this partition. Since in the cardinality expression in the function (12), $D = \mathit{Days}$, the expression, and therefore, by Pattern 3 and Proposition 17, the whole objective function (12) is fully value interchangeable.

To conclude this section on the value interchangeability of the constraints in the models of Section 3, we will look at the objective function of the warehouse location problem in Figure 8. By Pattern 5, we know that the cardinality expression is fully value interchangeable. *maintCost* is a constant and does not affect the symmetry. What remains is the summation function, where we, as opposed to Pattern 3, have references to elements of the domain in the summands. If we have:

Pattern 6:
cst $f : \mathit{Domain} \longrightarrow \mathbb{N}$
$\vdots$
$$\sum_{d:D \subseteq \mathit{Domain}} f(d)$$

of which the summation function in the warehouse location problem is an instance, we have (at least) two different partitions over which the function is

PVI. First, since it does not matter in which order the elements are added, any bijection on $\mathcal{D}_1 = \{D, \mathit{Domain} \setminus D\}$ will give the same solutions. But we could also make the partition $\mathcal{D}_2 = \{\{d : \mathit{Domain} \mid f(d) = f(d_i)\} \mid d_i \in \mathit{Domain}\}$. Which of these partitions we choose could be different in different cases. In the warehouse location problem though, the decision variable occurs in the subscript of the summation, which means that this would not help us partition the values. Instead we say that the function (and therefore the whole objective function in the model) is PVI over $\mathcal{D} = \{\{w : \mathit{Warehouses} \mid \forall s \in \mathit{Stores} \,.\, \mathit{supplyCost}(s, w) = \mathit{supplyCost}(s, w_i)\} \mid w_i \in \mathit{Warehouses}\}$.

6 Value Interchangeability in Compositions of Constraints

In Section 5 we saw how we could derive symmetries for single constraints and thus for problems with one constraint. In many CSPs, though, we have more than one constraint. In Section 6.1 we will look at how we can compositionally derive symmetries for problems with more than one constraint, and in Section 6.2 we study the semantics of some common constraints to see if we can derive a stronger symmetry than the general method of Section 6.1 would give us.

6.1 Composition of Constraint Symmetries

In Propositions 13 and 15, originally found in [4], we saw that given two CSPs $\mathcal{P}_1 = \langle V, D, C_1 \rangle$ and $\mathcal{P}_2 = \langle V, D, C_2 \rangle$, which are PVI over $\mathcal{D}_1$ and $\mathcal{D}_2$ respectively, their composition $\mathcal{P}_1 \wedge \mathcal{P}_2$ is PVI over $\mathcal{D} = \{D_1 \cap D_2 \mid D_1 \in \mathcal{D}_1 \wedge D_2 \in \mathcal{D}_2 \wedge D_1 \cap D_2 \neq \emptyset\}$. We observe that $\mathcal{P}_1 \wedge \mathcal{P}_2 \equiv \langle V, D, C_1 \wedge C_2 \rangle$ and that the proposition therefore can be applied to one CSP with two constraints. Using Proposition 15 repeatedly allows us always to derive some, but perhaps suboptimal, partition of the values over which a given CSP is PVI, as long as we know the partitions over which the constraints in the problem are PVI.

Now we can apply this proposition to our results from Section 5 on the symmetry of single constraints in the models of Section 3 to make conclusions on the symmetries of the whole models:

Progressive Party: In Section 5.1 we saw that the first constraint in the model of Section 3.1 of the progressive party problem is fully value interchangeable. By Section 5.2, we know that the same holds for the second constraint. But as the third constraint is only PVI, we can, using Proposition 15, conclude that the progressive party problem is PVI over the partition of the values $\mathcal{D} = \{\{h : \mathit{Hosts} \mid \mathit{spareCap}(h) = \mathit{spareCap}(h_i)\} \mid h_i \in \mathit{Hosts}\}$.

Scene Allocation: This is a COP, where both the constraint (by Section 5.1) and the objective function (by Section 5.3) are fully value interchangeable. Then by Proposition 20 this model is fully value interchangeable.

Social Golfers: In the model of Figure 5 of this problem, we have two constraints, which, by Sections 5.1 and 5.2, are both value interchangeable, which, by Proposition 13, means that so is the whole model.

Graph Colouring: Already in Section 5.2, we could conclude that this problem is fully value interchangeable, since its only constraint is.

BIBD: In the model in Figure 7, as mentioned earlier, the decision variable is not a function, but a relation between the *Varieties* and the *Blocks*. This is analogous to a function in $\mathit{Varieties} \times \mathit{Blocks} \longrightarrow \{0,1\}$. In the picture below, we see a solution to the BIBD problem where $v = b = 3$, $k = r = 1$ and $\lambda = 0$. Here we clearly see that the values 0 and 1 are not interchangeable, which means that we only have value interchangeability over the trivial partition. We will instead deal with the symmetries of this problem in the section on variable symmetries in single constraints, namely Section 8.

1	0	0
0	1	0
0	0	1

Warehouse Location: In the model of this problem, we have a constraint, which by Section 5.1 is PVI over $\mathcal{D}_1 = \{\{w : \mathit{Warehouses} \mid \mathit{cap}(w) = \mathit{cap}(w_i)\} \mid w_i : \mathit{Warehouses}\}$. We also have an objective function, which by Section 5.3 is PVI over $\mathcal{D}_2 = \{\{w : \mathit{Warehouses} \mid \forall s \in \mathit{Stores} \ . \ \mathit{supplyCost}(s,w) = \mathit{supplyCost}(s,w_i)\} \mid w_i \in \mathit{Warehouses}\}$. From this and Proposition 20 we can conclude that the whole model is PVI over the partition $\mathcal{D} = \{\{w : \mathit{Warehouses} \mid \mathit{cap}(w) = \mathit{cap}(w_i) \wedge \forall s \in \mathit{Stores} \ . \ \mathit{supplyCost}(s,w) = \mathit{supplyCost}(s,w_i)\} \mid w_i \in \mathit{Warehouses}\}$.

n-Queens: In the model of this problem, the first constraint matches Pattern 2 in Section 5.2, but the other two does not. The reason for this is that the problem, modelled this way, is only symmetric over the trivial partition of the values, i.e. $\mathcal{D} = \{\{d_i\}, d_i \in \mathit{Domain}\}$.

6.2 Aggregation

As observed in [4], the symmetries derived using Proposition 13 and 15 might be suboptimal. One way of dealing with this is by remodelling problems using aggregation, or at least recognising that a conjunction of constraints corresponds to some aggregated constraint, for which a stronger symmetry can be derived. Let us at first look at the global_cardinality constraint, available in SICStus

Prolog for instance, to see an example of a constraint that corresponds to a conjunction of count constraints.

$$\text{global\_cardinality}(Xs,\ Vals)$$

where $Xs = [x_1, \ldots, x_d]$ is a list of integers or decision variables, and $Vals = [k_1 - v_1, \ldots, k_n - v_n]$ is a list of pairs where each key k_i is a unique integer and v_i is a domain variable or an integer. It is true if every element of Xs is equal to some key, k_i, and for each pair $k_i - v_i$, exactly v_i elements of Xs are equal to k_i. This constraint is equivalent to:

dom Ks, Vs, Xs
cst $g : Ks \longrightarrow Vs$
var $f : Xs \longrightarrow Ks$
solve
$\quad \forall(k : Ks)\text{count}(g(k))(x : Xs \mid f(x) = k)$

This constraint we have already seen in Section 5.1 as an instance of Pattern 1, modified according to case (2g), so compared with any constraint of the type $\forall(\ldots)\text{count}(\ldots)(\ldots)$ it is not very general and it does not help us to derive any stronger symmetries than we have already done.

Let us instead look at a conjunction of constraints, all matching case 1:

$$\begin{array}{l}
\text{var } f : \mathit{Variables} \longrightarrow \mathit{Domain} \\
\vdots \\
\forall(d : D_1)\text{count}(\langle \mathit{Set} \rangle)(v : V_1 \mid f(v) = d) \\
\wedge\forall(d : D_2)\text{count}(\langle \mathit{Set} \rangle)(v : V_2 \mid f(v) = d) \\
\vdots \\
\wedge\forall(d : D_n)\text{count}(\langle \mathit{Set} \rangle)(v : V_n \mid f(v) = d)
\end{array}$$

where $D_i \subseteq \mathit{Domain}$, $V_i \subseteq \mathit{Variables}$ and where the $\langle \mathit{Set} \rangle$ in the count expression is the same set in all the constraints. Then by considering the semantics of these constraints, we see that this conjunction is PVI over $\mathcal{D} = \{D_1 \cup D_2 \cup \ldots \cup D_n, \mathit{Domain} \setminus D_1 \cup D_2 \cup \ldots \cup D_n\}$, whereas we, using the result of case 1 and Proposition 15 would get a much finer partition.

Example 25 Take the case where $n = 2$, $\mathit{Domain} = \{1, 2, 3, 4\}$, $D_1 = \{1, 2\}$ and $D_2 = \{2, 3\}$. Then using aggregation we derive PVI over the partition $\mathcal{D} = \{\{1, 2, 3\}, \{4\}\}$ whereas our previous results would only give us the trivial partition $\mathcal{D}' = \{\{1\}, \{2\}, \{3\}, \{4\}\}$.

7 Variable Interchangeability

In Section 3.5 we mentioned that the BIBD problem is variable symmetric, and as we will see, many of the other problems that we have discussed, apart from being (piecewise) value interchangeable, are also variable interchangeable. Before deriving these symmetries, we will start by transforming the definitions and propositions on value interchangeability of Section 4 into results about variable interchangeability, and also see how more complex symmetries can be derived compositionally. We will use the Definition 12 of [4] for variable symmetry and then derive definitions and propositions corresponding to Definitions 7 to 11 and Propositions 1 to 6 of that paper, but for variable symmetry. This means that Definitions and Propositions 27 to 43 are my translations of results from [4] on value symmetry to results on variable symmetry. In the following the corresponding definition/proposition of [4] is given in parenthesis after the number of the new definition/proposition. Note that the numbers given in parenthesis after these definitions and propositions does not, as opposed to in Section 4 indicate that these definitions and propositions are in [4], but only that they are analogous to the indicated definition/proposition. The Definitions and Propositions 26 and 44 to 50 are copied from [4]. Now, let us first repeat the definition for variable interchangeability (Definition 12 of [4]):

Definition 26 (Definition 12) Let $\mathcal{P} = \langle V, D, C\rangle$ be a CSP. $\mathcal{P}$ is *variable interchangeable* if, for each solution $\sigma \in Sol(\mathcal{P})$ and each bijection $b : V \longrightarrow V$, the function $\sigma \circ b \in Sol(\mathcal{P})$.

Example 27 The CSP $\mathcal{P} = \langle V, D, \text{allDifferent}(v_1, v_2, v_3)\rangle$ is variable interchangeable if $V = \{v_1, v_2, v_3\}$, else the next definition applies:

Definition 28 (Definition 7) Let $\mathcal{P} = \langle V, D, C\rangle$ be a CSP and $\mathcal{V}$ be a partition of V. $\mathcal{P}$ is *piecewise variable interchangeable* (PVarI) over $\mathcal{V}$ if, for each solution $\sigma \in Sol(\mathcal{P})$ and each piecewise-interchangeable bijection b (see Definition 5) over $\mathcal{V}$, $\sigma \circ b \in Sol(\mathcal{P})$.

Note that, if $\mathcal{P} = \langle V, D, C\rangle$ is variable interchangeable, then it is PVarI over $\{V\}$.

Example 29 Take the CSP $\mathcal{P}$ as in Example 27, i.e. $\mathcal{P} = \langle V, D, C\rangle$, where $C = \text{allDifferent}(v_1, v_2, v_3)$. Then $\mathcal{P}$ is piecewise variable interchangeable over $\mathcal{V} = \{\{v_1, v_2, v_3\}, V \setminus \{v_1, v_2, v_3\}\}$, i.e. if $V = \{v_1, v_2, v_3\}$, then $\mathcal{P}$ is PVarI over $\mathcal{V} = \{V\}$.

Definition 30 (Definition 8) A global function $f : (V \longrightarrow D) \longrightarrow \mathbb{N}$ is *variable interchangeable* if, for each assignment $\sigma : V \longrightarrow D$ and each bijection $b : V \longrightarrow V$, $f(\sigma) = f(\sigma \circ b)$.

Definition 31 (Definition 9) Let $\mathcal{V}$ be a partition of V. A global function $f : (V \longrightarrow D) \longrightarrow \mathbb{N}$ is *piecewise variable interchangeable* over $\mathcal{V}$ if, for each assignment $\sigma : V \longrightarrow D$ and piecewise-interchangeable bijection b over $\mathcal{V}$, $f(\sigma) = f(\sigma \circ b)$.

Example 32 Take the global function nbDistinct$(v_1, \ldots v_k) : (V \longrightarrow D) \longrightarrow \mathbb{N}$, where $V = \{v_1, v_2, \ldots, v_n\}$ and $k \leq n$. Then nbDistinct$(v_1, v_2, \ldots, v_k)$ is variable interchangeable if $k = n$, while on the other hand if $k < n$, it is PVarI over $\mathcal{V} = \{\{v_1, v_2, \ldots, v_k\}, \{v_{k+1}, v_{k+2}, \ldots, v_n\}\}$.

Definition 33 (Definition 10) Let $\mathcal{O} = \langle V, D, C, f\rangle$ be a COP. $\mathcal{O}$ is *variable interchangeable* if, for each solution $\sigma \in Sol(\mathcal{O})$ and each bijection $b : V \longrightarrow V$, $\sigma \circ b \in Sol(\mathcal{O})$ and $f(\sigma) = f(\sigma \circ b)$.

Definition 34 (Definition 11) Let $\mathcal{O} = \langle V, D, C, f\rangle$ be a COP and $\mathcal{V}$ be a partition of V. $\mathcal{O}$ is *piecewise variable interchangeable* over $\mathcal{V}$ if, for each solution $\sigma \in Sol(\mathcal{O})$ and each piecewise-interchangeable bijection b over $\mathcal{V}$, $\sigma \circ b \in Sol(\mathcal{O})$ and $f(\sigma) = f(\sigma \circ b)$.

Now, that we have defined variable interchangeability for CSPs as well as COPs, we can go on with the propositions for composition of symmetries.

Proposition 35 (Proposition 1) Let $\mathcal{P}_1 = \langle V, D, C_1\rangle$ and $\mathcal{P}_2 = \langle V, D, C_2\rangle$ be two variable interchangeable CSPs. Then, their composition $\mathcal{P}_1 \wedge \mathcal{P}_2$ is variable interchangeable.

Example 36 Take the two CSPs $\mathcal{P}_1 = \langle V, D, \text{allDifferent}(v_1, v_2, v_3)\rangle$ and $\mathcal{P}_2 = \langle V, D, v_1 + v_2 + v_3 = k\rangle$, where $V = \{v_1, v_2, v_3\}$. Then $\mathcal{P}_1 \wedge \mathcal{P}_2$ is variable interchangeable.

Proposition 37 (Proposition 2) Let $\mathcal{P}_1 = \langle V, D, C_1\rangle$ and $\mathcal{P}_2 = \langle V, D, C_2\rangle$ be two CSPs. Assume that $\mathcal{P}_i$ is piecewise variable interchangeable over partition $\mathcal{V}_i$ of V $(1 \leq i \leq 2)$. Then the composition $\mathcal{P}_1 \wedge \mathcal{P}_2$ is piecewise variable interchangeable over $\mathcal{V} = \{V_1 \cap V_2 \mid V_1 \in \mathcal{V}_1 \wedge V_2 \in \mathcal{V}_2 \wedge V_1 \cap V_2 \neq \emptyset\}$

Proof. Take $\sigma \in Sol(\mathcal{P}_1 \wedge \mathcal{P}_2)$, then $\sigma \in Sol(\mathcal{P}_1) \wedge \sigma \in Sol(\mathcal{P}_2)$. By the construction of $\mathcal{V}$, any bijection b which is piecewise interchangeable over $\mathcal{V}$ is also a piecewise interchangeable bijection over $\mathcal{V}_1$ and $\mathcal{V}_2$. Since $\mathcal{P}_i$ is PVarI over $\mathcal{V}_i$ we have $\sigma \circ b \in Sol(\mathcal{P}_1)$ and $\sigma \circ b \in Sol(\mathcal{P}_2)$, from which follows that $\sigma \circ b \in Sol(\mathcal{P}_1 \wedge \mathcal{P}_2)$, i.e. $\mathcal{P}_1 \wedge \mathcal{P}_2$ is PVarI over $\mathcal{V}$.

Note that, in Propositions 35 and 37, we have only considered CSPs with the same set of variables. Before looking at what happens when this is not the case, we shall look at the composition of function symmetries.

Proposition 38 (Proposition 3) Let f_1 and f_2 be two global functions of signature $(V \longrightarrow D) \longrightarrow \mathbb{N}$. If f_1 and f_2 are variable interchangeable, then so is $f_1 \star f_2$, where $\star \in \{+, -, \times\}$.

Example 39 Let $V = \{v_1, v_2, v_3\}$ and take the two variable interchangeable functions $f_1 = \text{nbDistinct}(v_1, v_2, v_3)$ and $f_2 = v_1 + v_2 + v_3$, then $3f_1 + 4f_2$ is variable interchangeable.

Proposition 40 (Proposition 4) Let f_1 and f_2 be two global functions of signature $(V \longrightarrow D) \longrightarrow \mathbb{N}$, and let f_1 and f_2 be PVarI over $\mathcal{V}_1$ and $\mathcal{V}_2$ respectively, then $f_1 \star f_2$ ($\star \in \{+, -, \times\}$) is PVarI over $\mathcal{V} = \{V_1 \cap V_2 \mid V_1 \in \mathcal{V}_1 \wedge V_2 \in \mathcal{V}_2 \wedge V_1 \cap V_2 \neq \emptyset\}$.

Proposition 41 (Proposition 5) Let $\mathcal{O} = \langle V, D, C, f\rangle$ be a COP and $\mathcal{P} = \langle V, D, C\rangle$. If $\mathcal{P}$ and f are variable interchangeable, then $\mathcal{O}$ is variable interchangeable. If $\mathcal{P}$ is PVarI over partition $\mathcal{V}_1$ of V and f is PVarI over partition $\mathcal{V}_2$ of V, then $\mathcal{O}$ is PVarI over $\mathcal{V} = \{V_1 \cap V_2 \mid V_1 \in \mathcal{V}_1 \wedge V_2 \in \mathcal{V}_2 \wedge V_1 \cap V_2 \neq \emptyset\}$.

Proposition 42 (Proposition 6) Let $f : (V \longrightarrow D) \longrightarrow \mathbb{N}$ be a global function and $\mathcal{V}$ be a partition of V. If f is PVarI over $\mathcal{V}$, then the CSP $\langle V, D, f \approx 0\rangle$ is PVarI over $\mathcal{V}$ as well, where $\approx \in \{>, \geq, =, \neq, \leq, <\}$.

Remark In many cases, when composing CSPs, it is possible to find a coarser partition over which the composition is PVarI, than we would using Proposition 37. Look at the following example:

Example 43 Take the CSPs $\mathcal{P}_1 = \langle V, D, v_1 = v_2\rangle$ and $\mathcal{P}_2 = \langle V, D, v_2 = v_3\rangle$, where $V = \{v_1, v_2, v_3\}$. Then $\mathcal{P}_1$ is PVarI over the partition $\mathcal{V}_1 = \{\{v_1, v_2\}, \{v_3\}\}$ and $\mathcal{P}_2$ is PVarI over $\mathcal{V}_2 = \{\{v_1\}, \{v_2, v_3\}\}$. Then by Proposition 37 $\mathcal{P}_1 \wedge \mathcal{P}_2$ is PVarI over $\mathcal{V} = \{V_1 \cap V_2 \mid V_1 \in \mathcal{V}_1 \wedge V_2 \in \mathcal{V}_2 \wedge V_1 \cap V_2 \neq \emptyset\}$, i.e. $\mathcal{P}_1 \wedge \mathcal{P}_2$ is PVarI over $\mathcal{V} = \{\{v_1\}, \{v_2\}, \{v_3\}\}$, but $\mathcal{P}_1 \wedge \mathcal{P}_2$ is in fact fully variable interchangeable, since if we have a solution such that $v_1 = v_2$ and $v_2 = v_3$, then $v_1 = v_3$, i.e. these three variables are fully interchangeable. To see this we need more information about the CSPs $\mathcal{P}_1$ and $\mathcal{P}_2$ than the partitions over which they are PVarI, though, such as in this example the knowledge that equality is transitive.

Our example above illustrates that even if it would be nice to be able to always derive PVarI over a partition with as few elements as possible, this sometimes requires more information than we have about the problem. The important thing is that we do not derive any symmetries that are not there, and we can conclude that if a CSP $\mathcal{P}_1$ is PVarI over $\mathcal{V} = \{V_1, \ldots, V_m\}$ then it is also PVarI over $\mathcal{V}' = \{W_1, \ldots, W_n\}$, where for all $1 \leq i \leq n$ we have $W_i \subseteq V_j$ for some $1 \leq j \leq m$. In particular, any CSP $\mathcal{P} = \langle\{v_1, v_2, \ldots, v_n\}, D, C\rangle$ is PVarI over the trivial partition of its variables, $\mathcal{V} = \{\{v_1\}, \{v_2\}, \ldots, \{v_n\}\}$.

Since in the relational models of the progressive party and the social golfers problem we use matrix models, we end this section by including some results on matrix models from [4], and again give the corresponding numbers of [4] of the definitions and propositions in parentheses. Formally, a matrix M of variables can be modelled as a bijection $X \times Y \to V$, where X are the row indices of M, Y its column indices, and V its set of variables. For clarity, we use traditional notations: $M[i, j]$ denotes the variable in row i and in column j, $M[i]$ row i, and $M[*, j]$ column j. We assume that all matrices are defined over row indices X and column indices Y.

Definition 44 (Definition 15) A *matrix-CSP* (MCSP) is a triple $\langle M, D, C\rangle$, where M is a matrix of variables, D denotes the set of values for these variables, and $C : (M \to D) \to Bool$ specifies which assignments of values to the variables are solutions. A *solution* to an MCSP $\mathcal{P} = \langle M, D, C\rangle$ is a function $\sigma : M \to D$ such that $C(\sigma) = true$. The set of solutions to $\mathcal{P}$ is denoted by $Sol(\mathcal{P})$.

An example of a matrix-CSP is the model of the progressive party problem in Figure 2. There we have a matrix of variables, i.e. the set $\{schedule(g, p) \mid g : Guests, p : Periods\}$. The domain is the set *Hosts* and the constraint is the conjunction of the three forall statements, which for any given assignment of values to the variables is either true or false.

The next definitions specify column interchangeability, a "global" form of variable interchangeability.

Definition 45 (Definition 16) A *column permutation* for a matrix M is a function $\rho : M \to M$ such that

$$M[i, j] = \rho(M)[i, b(j)] \quad (i \in X \ \wedge \ j \in Y)$$

for some bijection $b : Y \to Y$.

Definition 46 (Definition 17) An MCSP $\mathcal{P} = \langle M, D, C\rangle$ is *column interchangeable* if, for each solution $\sigma \in Sol(\mathcal{P})$ and each column permutation $\rho : M \to M$, the function $\sigma \circ \rho \in Sol(\mathcal{P})$.

Proposition 47 (Proposition 7) Let $\mathcal{P}_1 = \langle M, D, C_1\rangle$ and $\mathcal{P}_2 = \langle M, D, C_2\rangle$ be two column interchangeable MCSPs. Then, their composition $\mathcal{P}_1 \wedge \mathcal{P}_2$ is column interchangeable.

These results are also generalised to concern piecewise interchangeability.

Definition 48 (Definition 18) Let $\mathcal{Y}$ be a partition over Y. A *piecewise column permutation* over $\mathcal{Y}$ for a matrix M is a function $\rho : M \to M$ such that

$$M[i, j] = \rho(M)[i, b(j)] \quad (i \in X \ \wedge \ j \in Y)$$

for some piecewise-interchangeable bijection b over $\mathcal{Y}$.

Definition 49 (Definition 19) Let $\mathcal{Y}$ be a partition over Y. An MCSP $\mathcal{P} = \langle M, D, C\rangle$ is *piecewise column interchangeable* over $\mathcal{Y}$ if, for each solution $\sigma \in Sol(\mathcal{P})$ and each piecewise column permutation ρ over $\mathcal{Y}$, the function $\sigma \circ \rho \in Sol(\mathcal{P})$.

Proposition 50 (Proposition 8) Let $\mathcal{P}_1 = \langle M, D, C_1\rangle$ and $\mathcal{P}_2 = \langle M, D, C_2\rangle$ be two piecewise column interchangeable MCSPs over $\mathcal{Y}_1$ and $\mathcal{Y}_2$ respectively. Then, their composition $\mathcal{P}_1 \wedge \mathcal{P}_2$ is piecewise column interchangeable over

$$\mathcal{Y} = \{Y_1 \cap Y_2 \ \mid \ Y_1 \in \mathcal{Y}_1 \ \wedge \ Y_2 \in \mathcal{Y}_2 \ \wedge \ Y_1 \cap Y_2 \neq \emptyset\}.$$

These results naturally generalise to row interchangeability and also to concern matrix-COPS.

8 Variable Interchangeabililty in Single Constraints

As we saw in Section 7 variable interchangeability is quite similar to value interchangeability. The difference is that we look for interchangeability of the variables of some CSP, instead of at the interchangeability of the values. This means that had the values for which we looked at the interchangeability of in Section 5 been variables instead of values, we would have had some kind of variable interchangeability. This means that we could have used the same patterns as in section 5 to check for variable interchangeability if the elements of the domain that has some kind of symmetry are variables in the model. But, as we will see when looking at the models of Section 3 again, the patterns of Section 5 are not applicable to them, when looking for variable interchangeability, but we will have to derive new patterns that show us that some variables are interchangeable.

There is one thing that makes variable interchangeability more complicated than value interchangeability, though, and it is the case when we deal with matrix variables. If we have matrix variable interchangeability, or ordinary variable interchangeability depends on the decision variable. If the variable is a function in $Var \longrightarrow Domain$ and the elements of Var are (piecewise) interchangeable, then we have (piecewise) variable interchangeability. But if the variable on the other hand is a function in $Xs \times Ys \longrightarrow Domain$ and the elements of Xs/Ys are (piecewise) interchangeable, then we have (piecewise) row/column interchangeability. This means that we can start by checking if the values of some domain are interchangeable or not, and then by looking if we have a matrix model or not, see what type of symmetry we have. We state this in a proposition that shows the relation between value, variable and row/column interchangeability:

Proposition 51 Assume that in a model of a CSP $\mathcal{P} = \langle V, D, C \rangle$ some set S occurs and that for any piecewise interchangeable bijection b over the partition $\mathcal{S}$ of S, it is the case that for every occurrence of an element s from S in the model of $\mathcal{P}$, replacing s by $b(s)$ preserves the solutions. Then if S is the set of values of $\mathcal{P}$, i.e. if $S = D$, then $\mathcal{P}$ is piecewise *value* interchangeable over $\mathcal{S}$ or if $S = V$, then $\mathcal{P}$ is piecewise *variable* interchangeable over $\mathcal{S}$ or finaly if S is the set of rows/columns of $\mathcal{P}$, then $\mathcal{P}$ is piecewise *row/column* interchangeable over $\mathcal{S}$.

Let us start by looking at the progressive party problem. In the relational model of Figure 2, we have chosen to let the variable be a matrix, where the rows represents the *Guests* and the columns represents the *Periods*. Each element in the matrix, $[g, p]$, is given a value representing the host that guest g visits in period p. Now, the *Periods* are fully interchangeable in the model and this corresponds, in our relational model, to column interchangeability as defined in Definition 46. The only thing we have to do to be able to detect this in the model is to construct patterns that lets us derive that the *Periods* are fully interchangeable in all the constraints of the model in Figure 2, and then note

that we are dealing with a matrix model, where the *Periods* are represented by the columns in the decision variable.

Let us for example start with the following pattern:

$$\textbf{Pattern 7: } \underbrace{\forall(\langle \mathit{QuantExpr1} \rangle)}_{1} \underbrace{\text{count}(\langle \mathit{Set} \rangle)}_{2} \underbrace{(\langle \mathit{QuantExpr2} \rangle \mid f(s) = \ldots)}_{3} \quad (14)$$

Constraints of this pattern are found in the progressive party problem (both the first and second constraint in Figure 2), the scene allocation problem, both constraints of the social golfers problem, and the warehouse location problem. This constraint can, in the same way as we did in Section 5.1, be divided into three parts. Now we choose one of the domains of the model for which we check for interchangeability. Again, whether this interchangeability of the elements of some domain implies variable, matrix variable or value interchangeability of the model depends on the role of this domain in the model. In the progressive party problem for example, we both have *Guests* and *Periods*, where the first set is interchangeable over guests with the same crew size and the second set is fully interchangeable which means that we have a piecewise row interchangeability, but a full column interchangeability. Now let us suppose that we in the constraint 14 are looking for interchangeability for the elements of a set *Set* and that $s \in \mathit{Set}$. This means that we, just as in Section 5.1 can go through each part of the constraint and see how it affects the interchangeability, with respect to *Set*, of the whole constraint, assuming that we at first have full symmetry. In order not to confuse things, we assume that the elements of *Set* are the variables of the actual CSP, such as the *Stores* in the warehouse problem (Figure 8), but if the model of the CSP instead have a matrix variable, where the rows/columns represents the elements of *Set*, all the results in the rest of this section can just as well be considered to deal with row/column interchangeability. Once we have derived the symmetries of the different parts of a constraint we can use the following proposition, corresponding to Proposition 23 of Section 5.1 to derive the symmetry for the whole constraint:

Proposition 52 Let us suppose that we have a constraint, such that the first part is "PVarI" over $\mathcal{V}_1$, the second is "PVarI" over $\mathcal{V}_2$, and the third is "PVarI" over $\mathcal{V}_3$. Then the whole constraint is PVarI over $\mathcal{V} = \{V_1 \cap V_2 \cap V_3 \mid V_1 \in \mathcal{V}_1 \wedge V_2 \in \mathcal{V}_2 \wedge V_3 \in \mathcal{V}_3 \wedge V_1 \cap V_2 \cap V_3 \neq \emptyset\}$.

Now, let us look at the "variable interchangeability" of the different parts of the constraint 14.

1. If $\langle \mathit{QuantExpr1} \rangle$ does not have any references to elements in *Set* it does not affect the interchangeability of the constraint, with respect to *Set*. Otherwise if we have $\forall(\ldots s : S \ldots)$, where $S \subseteq \mathit{Set}$, this part is "PVarI" over $\mathcal{S} = \{S, \mathit{Set} \setminus s\}$. That $s : S \subseteq \mathit{Set}$ can be expressed in many ways, of which some are mentioned in case 1 of Section 5.1. Note that we here and in the rest of this section abuse the concept of variable interchangeability when we say that a part of a constraing is "variable interchangeable" or "PVarI" in a similar way as was done in Section 8.

2. Again, we have a situation similar to the one concerning value interchangeability in Section 5.1. If there are no references to elements from *Vars* in $\langle Set \rangle$ we have full symmetry for this part, otherwise we have the same possibilities as in case 2 of Section 5.1.

3. The case of $\langle QuantExpr2 \rangle$, and therefore its variable interchangeability, is the same as for $\langle QuantExpr1 \rangle$.

Once the symmetries for these three parts have been established, we can use Proposition 52 to derive the symmetry for the whole constraint. Let us now use these results to make some conclusions on the variable interchangeability of some constraints in our relational models of Section 3, taking the into account whether we have matrix models or not:

Progressive Party: In the first two constraints in Figure 2, both the *Guests* and the *Periods* are fully interchangeable. This means that we have full row and full column interchangeability in these two constraints.

Scene Allocation: The *Scenes* are fully variable interchangeable in the constraint of the model of the scene allocation problem.

Social Golfers: Both the *Players* and the *Weeks* are fully interchangeable in the model of Figure 5, which means that we have full row and column interchangeability.

Warehouse Location: The stores in the constraint in the model of Figure 8 are fully variable interchangeable.

For the BIBD problem, we already in Section 3 noted that the model of Figure 7 has some kind of variable symmetry. Again we have a matrix model where both the rows (the *Varieties*) and the columns (the *Blocks*) are fully interchangeable but since the decision variable is not a function, but a relation in this model, we have to make a variant of the pattern above.

$$\underbrace{\forall(\langle QuantExpr1 \rangle)}_{1}\underbrace{\text{count}(\langle Set \rangle)}_{2}\underbrace{(\langle QuantExpr2 \rangle \mid f(s) \wedge \ldots)}_{3} \tag{15}$$

Constraints of this type displays exactly the same row/column interchangeability as the constraint 14 does in a matrix model when cases 1–3 above are applied to the three parts of the constraint.

Now let us look at the variable interchangeability of the objective functions and the numerical constraints of the models in Section 3. We will again simply use our results from Section 5, but apply them to sets of variables rather than values, to see if the variables are interchangeable, and then by looking at the decision variable decide whether we have variable or matrix row/column interchangeability.

Let us start with the *Periods* in the third constraint in the model of Figure 2. Using Pattern 3 in Section 5.3 we see that the *Periods* are fully interchangeable

in this constraint, i.e. we have a case of column interchangeability. The *Guests* on the other hand are row interchangeable over partitions of *Guests* with the same *crewSize*, a result that follows from Pattern 6 of Section 5.3.

In the scene allocation problem in Figure 4, we have an objective function which is a summation function. By Proposition 40 this means that the function is PVarI over the same partition as the summands. Since we are looking for the interchangeability of the *Scenes*, the expression $fee(a)$ does not affect the symmetry of the function. Then we have a cardinality expression $\text{card}\{(d : \mathit{Days}) \mid \exists(s : \mathit{Scenes} \mid \mathit{casting}(a, s) \wedge \mathit{shoot}(s) = d)$ with the following pattern:

$$\text{card}\{\underbrace{(\ldots)}_{1} \mid \exists(\underbrace{v : V \subseteq \mathit{Vars}}_{2} \mid \underbrace{\langle \mathit{Expr1} \rangle \wedge \langle \mathit{Expr2} \rangle}_{3})\} \tag{16}$$

To derive the symmetry of this constraint, let us first construct the following useful pattern:

Pattern 8:
$\langle \mathit{Expr\ 1} \rangle \wedge \langle \mathit{Expr\ 2} \rangle \wedge \ldots \wedge \langle \mathit{Expr\ n} \rangle$

If the expression $\langle \mathit{Expr\ i} \rangle$ is "PVarI" over the partition $\mathcal{V}_i$ of the variables, then the whole conjunction is "PVarI" over the partition $\mathcal{V} = \{V_1 \cap V_2 \cap \ldots \cap V_n \mid V_i \in \mathcal{V}_i \wedge V_1 \cap V_2 \cap \ldots \cap V_n \neq \emptyset\}$.

In the first part of the expression 16, we have no references to any variables, and so it does not affect the symmetry. In the second part, we have again a situation similar to the one in case 1 of Section 5.1, i.e. it is "PVarI" over the partition $\mathcal{V} = \{V, \mathit{Vars} \setminus V\}$ of the variables. In the third part of the constraint we have a conjunction of expressions for which we can use Pattern 8 to derive the interchangeability of the variables. Now it is possible to use Proposition 52 to derive the interchangeability for the whole constraint.

If we again look at the summation function in the scene allocation model of Figure 4, we can from the results above conclude that it is PVarI over the partition $\mathcal{V} = \{\{s : \mathit{Scenes} \mid \forall a : \mathit{Actors}\ .\ \mathit{casting}(a, s) = \mathit{casting}(a, s_i)\} \mid s_i : \mathit{Scenes}\}$.

We will end this section by looking at the objective function in the model of the warehouse location problem. Here we have a sum, where the addend by Proposition 38 and the discussion of expression 16 is fully "variable interchangeable" whereas the augend, if we translate Pattern 6 of Section 5.3 to concern variable interchangeability, is "PVarI" over $\mathcal{V} = \{\{s : \mathit{Stores} \mid \forall w \in \mathit{Warehouses}\ .\ \mathit{supplyCost}(s, w) = \mathit{supplyCost}(s_i, w)\} \mid s_i \in \mathit{Warehouses}\}$. From Proposition 40 now follows that the whole objective function is PVarI over $\mathcal{V}$.

9 Variable Interchangeability in Compositions of Constraints

In this section we use Propositions 35, 37, 47 and 50 and the results of the previous section to derive the variable symmetries for the whole problems in the models of Section 3.

Progressive Party: In the previous section, we concluded that the *Periods* in the model in Figure 2 are fully variable interchangeable in all three constraints. By Proposition 47 we then have full column interchangeability. The *Guests* on the other hand are fully interchangeable in the first two constraints of the model, but in the third one they are only interchangeable over the partition $\mathcal{R} = \{\{g : Guests \mid crewSize(g) = crewSize(g_i)\} \mid g_i \in Guests\}$, which by Proposition 50 means that this model is row interchangeable over $\mathcal{R}$.

Scene Allocation: In our model of the scene allocation problem, in Figure 4, the *Scenes* are the variables. They are fully interchangeable in the constraint of the model, but in the objective function they are PVarI over $\mathcal{V} = \{\{s : Scenes \mid \forall a : Actors \,.\, casting(a, s) = casting(a, s_i)\} \mid s_i : Scenes\}$, by the previous section. We can now, using Proposition 37, derive that the whole model is PVarI over $\mathcal{V}$.

Social Golfers: Both the *Players* and the *Weeks* are fully intechangeable in both the constraints of the model in Figure 5, which, by Proposition 47, means that we have full row and full column interchangeability.

BIBD: Already in the previous section we concluded that we have full row, as well as full column interchangeability in the BIBD model of Figure 7.

Warehouse Location: In the model in Figure 8 of this problem, we have again full variable interchangeability in the constraint, but in the objective function the *Stores* are interchangeable over the partition $\mathcal{V} = \{\{s : Stores \mid \forall w \in Warehouses \,.\, supplyCost(s, w) = supplyCost(s_i, w)\} \mid s_i \in Warehouses\}$, which means that the whole model is PVarI over this partition.

n-Queens: As in Section 6.1 we can conclude that none of our patterns of Section 7 fully matches the last two constraints of Figure 10, and again we conclude that this is only quite natural, since the variables, as well as the values, of the model, the *Queens*, are only interchangeable over the trivial partition $\mathcal{V} = \{\{q_i\} \mid q_i \in Queens\}$

10 Conclusion

In this work we have seen that it is possible to detect symmetries in many CSPs modelled in a relational language. By analysing constraints and objective functions occurring in the sample models, and sometimes parts of constraints or functions, we have been able to detect common patterns for which symmetries can be derived. As well value as variable interchangeability have been detected by in this way finding patterns that shows us that the elements of some set occurring in a (part of a) constraint are fully or partially interchangeable. Once

the symmetries for single constraints had been detected, symmetries for complete CSPs and COPs could easily be derived compositionally, using the results on compositional derivation of symmetries in [4].

We have studied seven CSPs for which symmetries of different types and in various degree have been detected. One way of continuing the work would be to look at more sample models in order to find other patterns that displays symmetries as well. An other way would be to generalise the results found here so that they cover more types of constraints. In both cases there are at least two possible directions one could choose. The first one is to further exploit the approach of dividing a constraint into smaller parts, that like "buildingblocks" can be put together to form a constraint, and for which very general patterns can be constructed. The symmetry for a whole constraint can then be compositionally derived from the symmetries of the "buildingblocks". The other approach would be to find common constraints, or perhaps even aggregates of constraints, for which symmetries can be detected at once without using the compositional approach. The benefit of the first approach is that it would probably be possible to find very general patterns so that at least some symmetries can be derived for more or less any constraint. The problem is that the symmetries derived compositionally in this way might be suboptimal. If we on the other hand choose the other approach, we would probably find more optimal symmetries in some cases, but it might on the other hand be hard to find patterns for all possible constraints.

References

[1] P. Flener, J. Pearson, and M. Ågren. The Syntax, Semantics, and Type System of ESRA. Technical report, ASTRA group, April 2003. Available at `http://www.it.uu.se/research/group/astra/`.

[2] P. Flener, J. Pearson, and M. Ågren. Introducing ESRA, a relational language for modelling combinatorial problems. In M. Bruynooghe, editor, *LOPSTR'03: Revised Selected Papers*, volume 3018 of *LNCS*, pages 214–232. Springer-Verlag, 2004.

[3] P. Van Hentenryck, P. Flener, J. Pearson, and M. Ågren. Tractable symmetry breaking for CSPs with interchangeable values. In *Proceedings of IJCAI'03*, pages 277–282. Morgan Kaufmann, 2003.

[4] P. Van Hentenryck, P. Flener, J. Pearson, and M. Ågren. Compositional derivation of symmetries for constraint satisfaction. In J.-D. Zucker and L. Saitta, editors, *Proceedings of SARA'05*, LNCS. Springer-Verlag, 2005. Supersedes Technical Report `http://www.it.uu.se/research/reports/2004-022/`.