Symmetry-breaking as a Prelude to Implied Constraints: A Constraint Modelling Pattern

Alan M. Frisch, Christopher Jefferson, Ian Miguel
AI Group, Department of Computer Science
University of York
{frisch,caj,ianm}@cs.york.ac.uk

1 Introduction

Constraint programming has been very successful in tackling a wide variety of combinatorial problems in industry and academia. However, to apply constraint programming tools to a domain, the problem must be *modelled* as a constraint program. Constructing an effective model is difficult, requiring a great deal of expertise. In order to encode this expertise in an automated system, *patterns* [11] must be recognised in problems, in effective models and in the process of taking a problem and formulating it as an effective model. This report considers two important modelling issues: symmetry-breaking and the generation of implied constraints, techniques which can reduce search effort considerably. In particular, it highlights a pattern in the modelling process: the role played by symmetry-breaking in enabling stronger implied constraints to be derived.

To date the choice of symmetry-breaking constraints has been determined solely by how much symmetry is broken, weighed against the cost of adding the symmetry-breaking constraints themselves. This report demonstrates that a weaker symmetry-breaking scheme might sometimes be preferred to a stronger one because of the implied constraints that can be derived from it.

Symmetry-breaking can lead to the simplification and/or decomposition of complex global constraints, leading to a stronger final model. This report exemplifies this point by constructing a model for the the Balanced Incomplete Block Design problem. The model introduces incomplete static symmetry-breaking constraints from which strong implied constraints can be derived. This is contrasted to a complete (or more complete) dynamic symmetry-breaking approach in which the implied constraints cannot be introduced.

2 Background

An instance of the finite domain *constraint satisfaction problem* (CSP, [2]) consists of a triple $\langle X, D, C\rangle$, where X is a set of variables, D is a set of domains, and C is a set of constraints. Each $x_i \in X$ is associated with a finite domain $D_i \in D$ of potential values. A variable is *assigned* a value from its domain. A constraint $c \in C$, constraining variables $x_i, \dots, x_j$, specifies a subset of the Cartesian product $D_i \times \cdots \times D_j$ indicating mutually compatible variable assignments. A solution is an assignment to all elements of X that satisfies all the constraints.

A symmetry in a constraint satisfaction problem is a bijection that maps solutions to solutions and non-solutions to non-solutions. Symmetry in constraint programs can cause problems for an algorithm that searches a space of partial assignments due to redundancy in the search space. One of the most popular methods for reducing symmetry is to add to the model extra constraints, so-called *symmetry-breaking constraints* [1].

Implied constraints are logical consequences of the initial specification of the problem. Though adding an implied constraint to the specification of a constraint satisfaction problem does not change the set of solutions, it can reduce the amount of search the solver has to do. Symmetry-breaking constraints are not logical consequences of the initial specification and adding them to a model may reduce the set of solutions. However, breaking symmetry is often an important first step which helps to generate implied constraints.

3 Symmetry-breaking and the Derivation of Implied Constraints

The following illustrative example is taken from the implied constraint section of the Oz finite domain constraint programming tutorial [9][1]. The problem is to find 9 distinct non-zero digits, A to I, which satisfy the constraint:

$$\frac{A}{BC} + \frac{D}{EF} + \frac{G}{HI} \quad = \quad 1 \tag{1}$$

Note that here and throughout, BC is shorthand for $10 * B + C$, EF for $10 * E + F$ and HI for $10 * H + I$.

From (1), the following implied constraints might be added:

$$A \leq BC \tag{2}$$
$$D \leq EF \tag{3}$$
$$G \leq HI \tag{4}$$

Since these constraints are satisfied by every assignment[2] these constraints will never trigger any pruning.

Since the digits must be distinct, bounds can be established on each denominator as follows:

$$12 \leq BC \leq 98 \tag{5}$$
$$12 \leq EF \leq 98 \tag{6}$$
$$12 \leq HI \leq 98 \tag{7}$$

These inequalities do not propagate immediately, but may be useful during search. They are also useful in deriving the largest value that any of the fractions can take, as follows:

$$\frac{A}{BC}, \frac{D}{EF}, \frac{G}{HI} \leq \frac{9}{12} = \frac{3}{4} \tag{8}$$

Also from (5–7), the smallest denominator for each fraction is 12. Hence:

$$\frac{A + D + G}{12} > 1 \tag{9}$$

As the following two subsections show, by adding symmetry-breaking constraints to this model much stronger implied constraints can be derived. In the first subsection we introduce incomplete symmetry-breaking constraints and implied constraints that follow from them. In the second subsection we consider three models, each with complete symmetry-breaking constraints and implied constraints that follow from them. In the third subsection we show that, once implied constraints are added, the models with incomplete symmetry-breaking are easier to solve than the one with complete symmetry breaking.

3.1 Breaking Symmetry by Ordering the Fractions

From the commutativity and associativity of addition and the fact that the variables have identical domains, the sub-expressions of (1) — $\frac{A}{BC}$, $\frac{D}{EF}$, and $\frac{G}{HI}$ — are symmetrical. To break this symmetry, ordering constraints may be added as follows:

$$\frac{A}{BC} \leq \frac{D}{EF} \leq \frac{G}{HI} \tag{10}$$

[1]`http://www.mozart-oz.org/documentation/fdt/node31.html#section.propagators.fractions`

[2]A constraint that is satisfied by every assignment follows from the domains of the problem without considering the constraints.

It should be noted that this does not break the symmetry completely. For example, the assignment:

$$\frac{1}{29} + \frac{3}{87} + \frac{4}{56} \tag{11}$$

is symmetric to:

$$\frac{3}{87} + \frac{1}{29} + \frac{4}{56} \tag{12}$$

Since the first two fractions in the above are equal, both of these assignments are allowed by (10). Nevertheless, (10) allows the derivation of some powerful implied constraints, as will be shown.

First, (10) can be substituted into (1) as follows:

$$\begin{aligned} \frac{A}{BC} + 2\frac{D}{EF} &\leq 1 & (13)\\ 3\frac{A}{BC} &\leq 1 & (14)\\ 2\frac{D}{EF} + \frac{G}{HI} &\geq 1 & (15)\\ 3\frac{G}{HI} &\geq 1 & (16) \end{aligned}$$

In addition, from (10) upper and lower bounds can be derived for $\frac{D}{EF}$:

$$\frac{1}{8} \leq \frac{D}{EF} < \frac{1}{2} \tag{17}$$

The upper bound follows from the fact that if $\frac{D}{EF}$ were greater than a half, then $\frac{G}{HI}$ would also be greater than a half, with their sum exceeding 1. Similarly, if $\frac{D}{EF}$ were equal to a half, then $\frac{G}{HI}$ must equal a half, otherwise $\frac{D}{EF} + \frac{G}{HI}$ would exceed 1. In this case, $\frac{A}{BC}$ must equal 0, which is not possible with non-zero digits. The lower bound is derived from (8), from which it is clear that $\frac{A}{BC} + \frac{D}{EF} \geq \frac{1}{4}$. Similar reasoning to that employed for the upper bound then immediately yields the lower bound.

Arranging equations (14), (16) and (17) into linear form yields (18), (19) and (20–21), respectively.[3]

$$\begin{aligned} 3A &\leq BC & (18)\\ 3G &\geq HI & (19)\\ 2D &< EF & (20)\\ 8D &\geq EF & (21) \end{aligned}$$

Equations (18–21) are ternary, versus the original arity 9 constraint (1). Smaller arity constraints are more likely to be useful for pruning earlier in the search (since they depend on fewer variables being instantiated). In fact, enforcing the bounds consistency property [10] on (18–21) will prune values before search begins. Notice that these derived inequalities are *not* valid without the initial symmetry-breaking step. As an example, bounds reasoning on (7) and (19) gives:

$$G \geq 4, H \leq 2 \tag{22}$$

This immediately prunes almost half of the elements of the domain of G and all but two elements of the domain of H.

The implied constraints thus derived significantly reduce the search space of the fractions puzzle. This may not be the optimal model, however. As the following sub-section shows, different symmetry-breaking constraints leads to the derivation of different implied constraints, potentially reducing the search space even further.

[3]recall that $BC = 10B + C$, *etc.*

3.2 Breaking Symmetries by Lexicographic Ordering

We now consider three ways of adding complete symmetry-breaking constraints to the basic model. To begin, consider arranging the nine variables in a 3×3 matrix:

$$\begin{pmatrix} A & D & G \\ B & E & H \\ C & F & I \end{pmatrix}$$

Given the problem constraints, this matrix has column symmetry, i.e. any pair of columns can be exchanged in a (non-)solution to generate another (non-)solution. It does not have row symmetry. Therefore, all symmetry can be broken by lexicographically ordering the columns [4]. This can be done in six ways, depending on the order of significance of the variables in each column. Of the six alternatives, we consider three:

$$\langle A, B, C\rangle \leq_{\text{lex}} \langle D, E, F\rangle \leq_{\text{lex}} \langle G, H, I\rangle \tag{23}$$

$$\langle B, A, C\rangle \leq_{\text{lex}} \langle E, D, F\rangle \leq_{\text{lex}} \langle H, G, I\rangle \tag{24}$$

$$\langle C, A, B\rangle \leq_{\text{lex}} \langle F, D, E\rangle \leq_{\text{lex}} \langle I, G, H\rangle \tag{25}$$

We consider each of these in turn, producing three models, which we call LexA, LexB and LexC.

The LexA model uses constraint (23). This constraint together with the alldifferent constraint implies

$$A < D < G \tag{26}$$

From (26) and (9) it is easy to derive:

$$G > 5 \tag{27}$$

The LexB model uses constraint (24). This constraint together with the alldifferent constraint implies

$$B < E < H \tag{28}$$

This implies:

$$BC < EF < HI \tag{29}$$

By substituting (29) into (1) the following inequality can be derived:

$$\frac{A + D + G}{BC} > 1 \tag{30}$$

Re-arranging gives:

$$A + D + G > BC \tag{31}$$

Since the digits are all-different, the left-hand side can be at most $9 + 8 + 7 = 24$ in both cases. Hence, from (31):

$$B \leq 2 \tag{32}$$

The LexC model uses constraint (25). This constraint together with the alldifferent constraint implies

$$C < F < I \tag{33}$$

No strong constraints can be derived this constraint because C, F and I are the less significant digits in the denominators.

	Model				
	Basic	Fractions-breaking	LexA	LexB	LexC
Basic Composition	(1) AllDifferent$(A - I)$ (5–9)	Basic+ (10)	Basic+ (23)	Basic+ (24)	Basic+ (25)
Implied Constraints	n/a	(18–21)	(26) (27)	(28) (31)	(33)

Table 1: Composition of models of the Fractions Puzzle.

	Model			
	Fractions-breaking	LexA	LexB	LexC
No Implied Constraints	8,041	6,802	2,434	6,859
With Implied Constraints	2,507	5,641	734	5,736
Mean Improvement	2.39	1.30	2.29	1.19

Table 2: Mean choice points taken and mean improvement obtained when solving Fractions Puzzle over all variable orderings with and without implied constraints.

3.3 Comparison of Models

This sub-section compares the four models of the Fractions Puzzle: the one with incomplete symmetry-breaking presented in Section 3.1 and the three with complete symmetry-breaking presented in Section 3.2. In all cases BC, EF and HI are bound to an auxiliary variable to increase pruning. In addition, rational expressions are multiplied up to remove the need for the solver to deal with floating point expressions. The composition of each model is presented in Table 1.

Unsurprisingly, breaking symmetry gives a substantial reduction in the size of the search tree and overall search effort. Hence, the remainder of this section focuses on the four symmetry-breaking models. It is difficult to compare models meaningfully without considering the solution method adopted, in particular the variable and value ordering heuristics. For this reason the two models were compared using all 9! variable orderings of $\langle A, B, C, D, E, F, G, H, I\rangle$. To remove the effects of arriving at a good value ordering by chance, the entire search space is searched for each variable ordering.

Mean choice-point results are presented in Table 2. The implied constraints derived from the symmetry-breaking constraints lead to a visible reduction in search effort, highlighting the importance of implied constraint generation from symmetry-breaking. The most significant reduction is for the Fractions-breaking and LexB models. This is as expected, since the implied constraints in these two models are stronger than those derived for the LexA and LexC models.

A comparison between the Fractions-breaking model, which does not break symmetry completely, and the Lex models, which do, is presented in Table 3. As can be seen from the table, on average the Fractions-breaking model is worse than all the Lex models when no implied constraints are added. However, when compared with the LexA and LexC models, this situation is reversed when implied constraints are added. This result gives an existence proof to the claim that a weaker symmetry-breaking scheme is sometimes preferable to a stronger one when implied constraints are added to both. Hence, the point is underlined that when selecting symmetry-breaking constraints, it is important to consider the effect of both the symmetry-breaking constraints and the implied constraints that can be generated from them. The challenge for constraint modelling is to formulate rules for the selection of symmetry-breaking constraints that lead to the best implied constraints.

	Model		
	LexA	LexB	LexC
No Implied Constraints	1.04	3.56	1.05
With Implied Constraints	0.56	3.42	0.56
Reversal	0.62	0.02	0.46

Table 3: Geometric means over all variable orderings of the ratio of choice points Fractions-breaking:LexX (where $X \in \{A, B, C\}$), and proportion of variable orderings in which the Fractions-breaking model takes more choice points before adding implied constraints, but fewer afterwards ('Reversal').

4 Decomposing constraints via Symmetry-breaking: Balanced Incomplete Block Designs

The *Balanced Incomplete Block Design* problem (BIBD, see www.csplib.org problem 28) is introduced to illustrate a further benefit that symmetry-breaking can bring: to allow the decomposition or simplification of complex constraints. To the best of our knowledge the model described here is new to the literature.

The following is an English definition of the BIBD problem (from CSPLib).

> Given a 5-tuple of positive integers, $\langle v, b, r, k, \lambda\rangle$, arrange v distinct objects in b blocks such that each block contains k distinct objects, each object occurs in exactly r different blocks and every two distinct objects occur together in exactly λ blocks.

Despite its relative simplicity, the BIBD has important practical applications, such as cryptography and experimental design.

The most common model for this problem consists of a b-column, v-row matrix of 0/1 decision variables, where a '1' entry represents the decision to assign the object associated with a particular row to the block associated with a particular column. Each row is constrained to sum to r, each column is constrained to sum to k and the scalar product of each pair of rows is constrained to equal λ. This model has both symmetrical rows and symmetrical columns: exchanging any pair of rows or columns in a given (non-)solution produces another (non-)solution.

A simple way to break a lot of this symmetry is to impose the constraints that both the rows and the columns are in lexicographic order (lex^2 [4]). Consider the following lexicographically-ordered solution to the BIBD $\langle 7, 7, 3, 3, 1\rangle$. Here, the columns are lexicographically ordered left to right with the most significant bit at the top of the column, and the rows are lexicographically ordered top to bottom, with the most significant bit at the left:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Notice that the first row is a sequence of 0's followed by a sequence of 1's. Since the elements of the first row are the most significant bits in the lexicographically ordered columns, the values assigned to the first row must, in any solution, be monotonically increasing from left to right. Hence, the first row comprises $b - r$ 0's followed by r 1's. A similar argument can be applied to the first column. Therefore, following lex^2 symmetry-breaking, the values of first row and first column of a BIBD can be fixed. This immediately reduces the search space by 2^{b+v-1} and leads to further improvements to the model, as discussed below.

4.1 Model Simplification and Constraint Decomposition

Consider the scalar product constraint between the first row and all other rows. Since, following symmetry-breaking, it is known that the final r elements of the first row are 1, to satisfy the scalar product constraint the last r positions of every other row must sum to λ. This can be seen in the example above, where $\lambda = 1$. Adding this simple sum constraint to each row except the first has the following consequences:

- The scalar product constraint between the first and all other rows can be removed entirely.
- The constraint in the original specification that each row must sum to r can be *decomposed* for every row except the first (which is fixed anyway). Having stipulated that the last r elements must sum to λ, the first $b - r$ elements must sum to $r - \lambda$. This leads to stronger pruning on the rows.

It is now possible to fix the second row also. First, the sum of the first $(b - r)$ entries of the second row is $(r - \lambda)$. Second, since the columns are lexicographically ordered, and the most significant digits of the first $(b - r)$ columns is equal (to 0), the first $(b - r)$ entries of the second row must be $(b - 2r + \lambda)$ 0's followed by $(r - \lambda)$ 1's. Similarly, the last r entries of the second row must be $(r - \lambda)$ 0's followed by λ 1's. This can be seen in the example above and results in a further 2^{b-1} reduction in the search space.

Of course, having fixed the second row, the scalar product constraint between this and subsequent rows can be replaced by decomposing the sum constraint using a similar argument to the one given above. The sum constraints on the third and all subsequent rows are now decomposed into four parts:

1. Entries 1 to $(b - r)$ must sum to $(r - \lambda)$. (the first row is fixed to 0 here).
2. Entries $(b - r + 1)$ to b must sum to λ. (the first row is fixed to 1 here).
3. Entries 1 to $(b - 2r + \lambda)$ and $(b - r + 1)$ to $(b - \lambda)$ must sum to $(r - \lambda)$ (the second row is fixed to 0 here).
4. Entries $(b - 2r + \lambda + 1)$ to $(b - r)$ and $(b - \lambda + 1)$ to b must sum to λ. (the second row is fixed to 1 here).

Note that the first entry is fixed in each case, since it is part of column one. This assignment will be propagated automatically by constraints (1) and (3) above.

4.2 Experimental Results

To test the utility of the reformulated model of the BIBD problem, experiments were performed on sixteen instances, as presented in Table 4. Three models were tested in all. Model A is the basic 0/1 variable model with lexicographic ordering on both the rows and columns. Model B is model A with the first row and first column set, as described above. Finally, to model B model C adds the reformulations described in Section 4.1.

A row-wise variable ordering is used in all cases branching on 0 before 1. It is infeasible to explore all orderings for this problem, and the row-wise 0/1 ordering has been shown previously to be a good basic ordering for BIBD models in which symmetry is broken by lexicographic ordering constraints [5].

Model B provides a modest, but measurable, improvement over Model A. This is unsurprising: the variable/value ordering will ensure that the first row is always assigned consistently with the problem and symmetry-breaking constraints at the top of the search tree. Most benefit, therefore, comes from propagation derived from setting the first column. In comparison, a substantial improvement is evident when using the fully reformulated Model C. As might be expected, the improvement is most significant when b is large relative to v. In this case, a larger proportion of the variables are set prior to search, and a larger proportion of the constraints are decomposed.

BIBD	Model A		Model B		Model C	
v, b, r, k, λ	choices	time	choices	time	choices	time
7,7,3,3,1	20	0.01	7	0.01	6	0.01
6,10,5,3,2	33	0.02	16	0.01	10	0.01
6,20,10,3,4	106	0.25	64	0.02	20	0.01
7,35,15,3,5	339	0.09	242	0.07	48	0.02
6,50,25,3,10	1,016	0.39	845	0.30	56	0.03
12,22,11,6,5	3,112	0.59	3,065	0.59	615	0.14
10,30,9,3,2	3,056	1.00	2,981	0.90	1,048	0.31
14,26,13,7,6	18,726	4.38	18,667	4.28	3,763	0.89
15,45,24,8,12	11,557	5.15	11,432	4.84	2,566	1.04
7,140,60,3,20	17,234	84.7	16,252	71.5	348	1.62
8,56,28,4,12	117,485	101.4	117,291	91.1	3,332	2.34
8,98,49,4,21	3,245,150	7,398.2	3,244,644	6,017.5	35,971	25.3
7,210,60,2,10	6,561	227.6	4,983	130.3	2,553	49.4
9,60,20,3,5	289,619	431.2	289,277	378.3	50,748	63.3
15,75,40,8,20	283,615	594.1	283,605	539.1	42,811	74.2
9,90,40,4,15	-	>12h	-	>12h	722,695	1785.5

Table 4: Time expended (in seconds) and number of choice points explored in finding one solution in each of 16 instances of the BIBD problem.

5 Static versus Dynamic Symmetry-breaking

Thus far, this paper has assumed that symmetry-breaking is performed by adding constraints to the model prior to search. It is worth considering how dynamic symmetry-breaking methods, such as SBDS [8] or SBDD [3] might also support implied constraint generation. These systems dynamically add constraints or examine the search tree to prevent areas of the search symmetrical to those already explored from being explored themselves. This immediately poses a problem for the static derivation of implied constraints, since there are no statically-posted symmetry-breaking constraints from which to make derivations.

Reconsider the Fractions puzzle in Section 3 and assume that the fractions are taken as the symmetrical objects, as in Section 3.1 but without the ordering given in (10). Now, since $\frac{G}{HI}$ is not distinguished from the other two fractions, it is not possible to derive an implied constraint as strong as, say, (22). At most, it is possible to say that one of A, D and G must be greater than or equal to 4, but this is clearly weaker than (22).

It may be possible to combine static and dynamic symmetry-breaking such that the static symmetry-breaking supports the derivation of useful implied constraints and the dynamic symmetry-breaking removes the remainder of the symmetry. However, typically it is very difficult to describe concisely the effect on a symmetry group, upon which the latest versions of SBDS and SBDD depend, of adding static constraints to break symmetry partially.

Constraint simplification/decomposition, as exemplified by the BIBD problem in the previous section, is also more difficult with dynamic symmetry-breaking for the same reason: it depends on static derivations that cannot, typically, be made. Again, approximations are the best that is possible. Section 4.1 described how each row of a BIBD can be broken into four parts. This remains true even if the location of the four parts is unknown. Hence, given two rows, x and y, for each pair of entries, x_i and y_i, it is possible to introduce an auxiliary variable a_{xyi} that records which of the four parts this pair belongs to. Occurrence constraints on a_{xy} would then ensure the correct number of members of each part. This, however, is considerably more expensive to maintain than the decomposition presented in Section 4.1.

6 Conclusion

The importance of both symmetry-breaking and the generation of implied constraints in formulating an effective model of a constraint satisfaction problem is well documented [****]. Symmetries can typically be broken in a number of ways, encompassing complete/incomplete static/dynamic approaches. Previously, the tradeoff has been the overhead of the symmetry-breaking scheme weighed against the benefit obtained. This report has demonstrated that, to obtain the best model, both the symmetry-breaking method and the implied constraints that can be generated from it must be considered.

To make full use of this result, it is necessary to be able to determine statically with some confidence when one model is better than another. Furthermore, automatic methods of generating implied constraints, such as CGRASS [7], must be tailored to expoit symmetry-breaking constraints to the greatest extent possible.

Acknowledgements

We thank Ian Gent, Warwick Harvey and Barbara Smith for their helpful comments.

References

[1] J. Crawford, M.L. Ginsberg, E. Luks, A. Roy. Symmetry-breaking predicates for search problems. In *Proceedings of KR'96: Principles of Knowledge Representation and Reasoning*, pp. 148–159, 1996.

[2] R. Dechter. *Constraint Processing.* Morgan Kaufmann, 2003.

[3] T. Fahle, S. Shamberger, M. Sellman. Symmetry breaking. In *Proceedings of the 7th International Conference on Principles and Practice of Constraint Programming*, pp. 93–107, 2001.

[4] P. Flener, A.M. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, J. Pearson, T. Walsh. Breaking Row and Column Symmetries in Matrix Models. *Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming*, pp. 462–476, 2002.

[5] A.M. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, T. Walsh. Global Constraints for Lexicographic Orderings *Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming*, pp. 93–108, 2002.

[6] A.M. Frisch, I. Miguel, T. Walsh. Extensions to Proof Planning for Generating Implied Constraints. *Proceedings of the Ninth Symposium on the Integration of Symbolic Computation and Mechanized Reasoning (Calculemus 01)*, pp. 130–141, 2001.

[7] A.M. Frisch, I. Miguel, T. Walsh. CGRASS: A System for Transforming Constraint Satisfaction Problems. In *Proceedings of the Joint Workshop of ther ERCIM Working Group on Constraints and the CologNet area on Constraint and Logic Programming on Constraint Solving and Constraint Logic Programming* (LNAI 2627), pp. 15–30, 2003.

[8] I.P. Gent, B.M. Smith. Symmetry breaking during search in constraint programming. In *Proceedings of the 14th European Conference on Artificial Intelligence*, pp 599–603, 2000.

[9] C. Schulte, G. Smolka. *Finite Domain Constraint Programming in Oz. A Tutorial.* `http://www.mozart-oz.org/documentation/fdt/index.html`.

[10] P. van Hentenryck, V. Saraswat, Y. Deville. Design, implementation and evaluation of the constraint language cc(FD). *Journal of Logic Programming*, 37 (1–3), pp. 139–164, 1998.

[11] T. Walsh. Constraint Patterns. *Proceedings of the 9th International Conference on Principles and Practice of Constraint Programming*, pp. 53–64, 2003.