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, UK
{frisch,caj,ianm}@cs.york.ac.uk

Abstract. Finite-domain constraint programming can be used to solve a wide range of problems by first *modelling* the problem as a set of constraints that characterise the problem's solutions, then searching for solutions that satisfy the constraints. Experts often augment models with implied constraints and constraints that break symmetries in the model. An emerging *pattern* in the modelling process, highlighted and demonstrated here, is that some powerful implied constraints can be derived only after symmetry-breaking constraints have been added. Furthermore, the choice between alternative symmetry-breaking constraints is commonly made by considering either the amount of symmetry broken or the strength of pruning obtained in comparison with the overhead of enforcing the constraints. We demonstrate that the choice should also consider the strength of the implied constraints derivable from the symmetry breaking constraints. We also discuss future automation of the selection of symmetry-breaking constraints and the derivation of implied constraints.

1 Introduction

Finite-domain constraint programming has been very successful in tackling a wide variety of combinatorial problems in industry and academia. A problem is solved with constraint programming by characterising, or *modelling*, the problem's solutions by a set of constraints and then searching for solutions to the constraints. Problems can usually be modelled in many ways, some of which may be much easier to solve than others. Constructing an effective model is difficult, requiring a great deal of expertise. This creates a modelling bottleneck, preventing widespread access to this powerful technology. Our long-term goal is to reduce this bottleneck by encoding modelling expertise in an automated modelling system. To perform this encoding, *patterns* must be identified in problems, in effective models [5, 19], and in the process of taking a problem and formulating it as an effective model.

This paper considers two important modelling issues: symmetry breaking and generating implied constraints, techniques that can reduce search effort considerably. It identifies a pattern in the modelling process: the role played by symmetry breaking in enabling stronger implied constraints to be derived. Though models that result from implicitly applying this pattern have been published [12, 16, 18], this paper presents what we believe to be the first explicit examination of the process of constructing such a model.

We begin by demonstrating, via a case study, the importance of this pattern in reducing search dramatically. Further, it has become increasingly difficult for a modeller to select from a growing number of symmetry-breaking methods [1, 2, 4, 7, 9, 10, 14]. This choice is often determined only by how much symmetry is broken or the strength of pruning obtained, versus the cost of adding the symmetry-breaking method [13, 15]. We show, via two further case studies, that this strategy does not always produce the best model. To obtain the best model, a symmetry-breaking scheme must be chosen not simply on its own merits, but also on the strength of the implied constraints derivable from it. This point is illustrated by demonstrating that a weaker symmetry-breaking scheme can outperform a stronger one due to the constraints that follow from it. Reflecting on these case studies, we consider how selecting a symmetry-breaking method, and the consequent derivation of implied constraints, can be encoded in informal rules as a step towards automation.

2 Background

A finite domain *constraint satisfaction problem* (CSP, [3]) consists of a finite set of decision variables, each with an associated finite domain of potential values, and a finite set of constraints. Each constraint specifies allowed combinations of assignments of values to a subset of the variables. An *assignment* gives each variable a value from its domain, and it is a *solution* if it satisfies all the constraints.

A symmetry in a constraint satisfaction problem is a bijection on the set of assignments that maps solutions to solutions—and, hence, non-solutions to non-solutions. Symmetry in a constraint program introduces redundancy in the search space of partial assignments. One method for reducing symmetry, and hence removing redundant parts of the search space, is to add to the model extra constraints, so-called *symmetry-breaking constraints* [2]. Another is to modify the search procedure to avoid visiting parts of the search space symmetric to those already explored [1, 4, 14].

Implied constraints are logical consequences of the constraints in a CSP. Though adding an implied constraint to 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. Thus, there are often constraints that are implied by a model only after the addition of symmetry-breaking constraints.

3 Symmetry Breaking and the Derivation of Implied Constraints

We first illustrate the basic modelling process pattern under consideration: implied constraints derived from symmetry-breaking constraints can drastically improve a model. Case study 1 is the *Balanced Incomplete Block Design* problem (BIBD, see www.csplib.org problem 28). To the best of our knowledge the reformulated model described here is new.

The BIBD is defined as follows: Given a 5-tuple of positive integers, $\langle v,\ b,\ r,\ k,\ \lambda\rangle$, assign each of v objects to 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 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 in row i, column j represents the decision to assign the ith object to the jth block. 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 row and column symmetry: exchanging a pair of rows or columns in a solution produces a solution.

A simple way to break much of this symmetry is to impose the constraints that both the rows and the columns are in lexicographic order (lex^2 [6]). We will refer to the matrix model with lex^2 symmetry breaking as the 'basic' model. 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:

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

We now consider the constraints implied by choosing lex^2 symmetry breaking. Notice in the above example 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 non-decreasing 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, the values of the first row and first column of a BIBD can be fixed.

Now consider the scalar product constraint between the first row and all other rows. Following symmetry breaking, it is known that the final r elements of the first row are 1, so 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, 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 that each row must sum to r can be *decomposed* for every row except the first (which is fixed). Since the last r elements sum to λ, the first $b-r$ elements must sum to $r-\lambda$. This leads to stronger pruning on the rows.

Row 2 can now be fixed. First, the sum of the first $(b-r)$ entries of row 2 is $(r-\lambda)$. Second, the columns are lexicographically ordered, and the most significant digits of the first $(b-r)$ columns are equal (to 0), so the first $(b-r)$ entries of row 2 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.

Of course, having fixed the second row, the scalar product constraint between this and subsequent rows can be replaced by decomposing the sum constraint. The sum constraints on the third and all subsequent rows are now decomposed thus:

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

Table 1 reports experiments to find one solution (finding all is infeasible) on 16 instances with the basic and reformulated models. In both models, lex^2 is imposed by a lexicographic-ordering constraint [7] between all adjacent rows and columns. The experiments were run on a 768Mb, 1GHz Windows-based Pentium III system. Here and throughout Ilog Solver 5.3 is used. Variable ordering is row-wise, top to bottom, left to right, branching on 0 first; this ordering has previously proven to be a good basic ordering for BIBD models with lex^2 symmetry breaking. [7].

BIBD	Basic Model		Reformulated Model	
v,b,r,k,λ	Choices	Time	Choices	Time
7,7,3,3,1	20	0.01	6	0.01
6,10,5,3,2	33	0.02	10	0.01
6,20,10,3,4	106	0.25	20	0.01
7,35,15,3,5	339	0.09	48	0.02
6,50,25,3,10	1,016	0.39	56	0.03
12,22,11,6,5	3,112	0.59	615	0.14
10,30,9,3,2	3,056	1.00	1,048	0.31
14,26,13,7,6	18,726	4.38	3,763	0.89
15,45,24,8,12	11,557	5.15	2,566	1.04
7,140,60,3,20	17,234	84.7	348	1.62
8,56,28,4,12	117,485	101.4	3,332	2.34
8,98,49,4,21	3,245,150	7,398.2	35,971	25.3
7,210,60,2,10	6,561	227.6	2,553	49.4
9,60,20,3,5	289,619	431.2	50,748	63.3
15,75,40,8,20	283,615	594.1	42,811	74.2
9,90,40,4,15	—	>12h	722,695	1785.5

Table 1. Time(s), choices made by Solver in finding 1 solution.

A substantial improvement is evident when using the reformulated model. 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. Experiments were also performed on the basic model with the first two rows and first column pre-set (omitted for space reasons). A much more modest improvement was obtained, emphasising the importance of performing as much inference as possible following symmetry breaking.

4 Extending the Pattern

Having established the utility of adding implied constraints following symmetry breaking, this section extends this basic pattern by considering how implied constraints should be considered in the selection of a symmetry-breaking scheme.

4.1 The 3-fractions Puzzle

Case study 2 is an instance of the n-fractions puzzle (CSPLib problem 41). This instance was chosen since it admits powerful implied constraints that vary according to how symmetry is broken while being small enough to search exhaustively. The goal is to find 9 distinct non-zero digits, A to I satisfying:

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

Here and throughout, BC is shorthand for $10 * B + C$, *etc.*

Given distinct digits, the following bounds are implied:

$$12 \leq BC \leq 98, \quad 12 \leq EF \leq 98, \quad 12 \leq HI \leq 98 \quad (2)$$

$$\frac{A}{BC}, \frac{D}{EF}, \frac{G}{HI} \leq \frac{9}{12} = \frac{3}{4} \quad (3)$$

Substituting (2) into (1), and given that the denominators must be distinct, the following can be derived:

$$A + D + G > 12 \quad (4)$$

Adding symmetry-breaking constraints to this model supports much stronger implied constraints. From the commutativity and associativity of addition and the fact that the variables have identical domains, the sub-expressions of (1) — A/BC, D/EF, and G/HI — are symmetrical. To break this symmetry, ordering constraints may be added:

$$\frac{A}{BC} \leq \frac{D}{EF} \leq \frac{G}{HI} \quad (5)$$

This model, which we will call 'Frac-breaking', does not break the symmetry completely. For example, $1/29 + 3/87 + 4/56$ and $3/87 + 1/29 + 4/56$ are symmetrical, but both satisfy (5) since the first two fractions are equal. Nonetheless, (5) allows the derivation of some powerful implied constraints.

Clearly, at least one of the fractions is less than or equal to $1/3$. Hence, from (5), $A/BC \leq 1/3$. By similar reasoning, $G/HI \geq 1/3$. Upper and lower bounds can also be derived for D/EF: $1/8 \leq D/EF < 1/2$. The upper bound follows because if $D/EF \geq 1/2$, then $G/HI \geq 1/2$ and $A/BC \leq 0$, which is not possible with non-zero digits. (3) implies $A/BC + D/EF \geq 1/4$. The lower bound follows. Arranging these inequalities into linear form gives:

$$3A \leq BC, \quad 3G \geq HI, \quad 2D < EF, \quad EF \leq 8D \quad (6)$$

Simple bounds reasoning on (2) and (6) now gives, for example, $G \geq 4$ and $H \leq 2$ prior to search.

We now discuss alternative symmetry-breaking constraints that lead to the derivation of different implied constraints, potentially reducing the search space even further. Consider arranging the 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: any pair of columns can be exchanged in a solution to generate a solution. It does not have row symmetry, so all symmetry can be broken by constraining the columns to be lexicographically ordered [6]. This can be done in six ways, depending on the order of significance of the variables in each column. We consider three alternatives:

$$\langle A, B, C\rangle \leq_{\text{lex}} \langle D, E, F\rangle \leq_{\text{lex}} \langle G, H, I\rangle \quad (7)$$

$$\langle B, C, A\rangle \leq_{\text{lex}} \langle E, F, D\rangle \leq_{\text{lex}} \langle H, I, G\rangle \quad (8)$$

$$\langle C, A, B\rangle \leq_{\text{lex}} \langle F, D, E\rangle \leq_{\text{lex}} \langle I, G, H\rangle \quad (9)$$

Model LexA uses (7). This and AllDifferent$(A-I)$ gives:

$$A < D < G \quad (10)$$

From (4) and AllDifferent$(A-I)$, one of $\{A, D, G\}$ is greater than 5. Hence, from (10):

$$G > 5 \quad (11)$$

Model LexB uses (8). This and AllDifferent$(A-I)$ gives:

$$B < E < H \quad (12)$$

This implies $BC < EF < HI$. Substituting BC for EF and HI in (1):

$$A + D + G > BC \quad (13)$$

Since the digits are distinct, the left-hand side can be at most $9 + 8 + 7 = 24$. Hence, from (13), $B \leq 2$.

Model LexC uses (9). This and AllDifferent$(A-I)$ gives:

$$C < F < I \quad (14)$$

No strong constraints can be derived from (14), because C, F and I are the less significant digits in the denominators.

Now we compare the symmetry-breaking models of the 3-fractions puzzle. The 'base' model, from which they are derived, consists of (1–4) and AllDifferent$(A-I)$. An auxiliary variable is introduced and constrained to be equal to each of BC, EF and HI to increase pruning. Rational expressions are multiplied out to avoid rounding errors. Table 2 gives the composition of the models tested.

	Frac-breaking	LexA	LexB	LexC
Initial Composition	Basic+(5)	Basic+(7)	Basic+(8)	Basic+(9)
Best/Worst Choices	910/46,211	895/33,947	253/20,462	849/30,294
Mean Choices	8,041	6,802	2,434	6,859
Implied Constraints	(6)	(10), (11)	(12),(13)	(14)
Best/Worst Choices	668/8,587	812/23,312	198/2,690	734/22,382
Mean Choices	2,507	5,641	734	5,736
Mean Improvement	2.39	1.30	2.29	1.19

Table 2. Composition of & comparison among models of 3-fractions over all variable orderings. Times (omitted: differing hardware platforms needed for 9! experiments) scale with choices.

For a thorough comparison, the 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 ordering.

Table 2 reports the number of choices reported by Solver. Individual improvements obtained per model by adding constraints implied by symmetry breaking underlines the importance of this modelling pattern. Further, if the models are ranked before and after implied constraints are added, the

Fractions-breaking model moves from worst to second best. This illustrates our claim that weaker symmetry breaking can sometimes outperform a stronger scheme when implied constraints are added. Therefore, when choosing a symmetry-breaking scheme, it is important to consider the implied constraints that can be derived.

4.2 Sports Scheduling with Odd Teams

Case study 3 considers models of the sports scheduling problem with an odd number of teams, and shows that a weaker set of symmetry-breaking constraints leads to a better model because of the implied constraints that it enables. The problem is a version, previously studied [10], of CSPLib prob026: Given an odd natural number n, find a schedule of n weeks such that each of n teams plays against each other exactly once and each team plays at most once in each week. Each week is divided into $(n-1)/2$ periods during which exactly one match takes place. Over the course of the schedule each team is to play exactly once in each period.

Our base model employs a 2-d matrix, *schedule*, in which the rows are indexed by the *weeks* and the columns by the *periods*. Each element of the matrix is a pair of decision variables, each having the set of *teams* as its domain. A constraint stipulates that the $n-1$ variables in each row of *schedule* are all-different. An occurrence constraint specifies that each *team* is assigned to exactly two of the $2n$ variables in each column. To ensure that each pair of teams plays exactly once we would like to impose an all-different constraint over all the pairs in *schedule*. Since typical implementations of all-different operate on a set of variables, but not a set of variable-pairs, we channel each pair of variables in *schedule* to a new variable whose domain has one element per pair of teams. Then we impose an all-different constraint over all the new variables.

Each element of *schedule* constrains a pair of indistinguishable variables that are to be assigned different teams (a team cannot play itself). So, for each element of *schedule* we assert that the team assigned to the first variable is strictly less than the team assigned to the second variable—thus breaking the symmetry within each pair of variables.

This base model still has value symmetry and row and column symmetry in *schedule*. We shall ignore the former and focus on two methods of dealing with the row and column symmetry. In doing this we treat each row and each column of *schedule*, which is of the form $[\langle X_1, Y_1\rangle, \langle X_2, Y_2\rangle, \ldots]$, as a vector of the form $[X_1, Y_1, X_2, Y_2 \ldots]$. The two alternative models, LEX and MS, differ in the constraints they use to break the row and column symmetry. LEX and MS impose symmetry-breaking constraints SBL and SBM respectively:

SBL: Columns are non-decreasing in the lexicographic ordering. Rows are non-decreasing in the lexicographic ordering.

SBM: Columns are non-decreasing in the lexicographic ordering. Rows are non-decreasing in the multiset ordering.

These ordering constraints are achieved by imposing (non-strict) lexicographic or multiset ordering constraints between pairs of adjacent rows or columns [7, 10]. Neither model dominates the other. Consider the following pair of partial assignments for the *schedule* matrix in the 3-teams instance, with the domains of uninstantiated variables denoted as sets. With the domains shown in (a), SBL is generalised arc-consistent but SBM is not; the opposite is true with (b).

$$(a)\ \begin{pmatrix} \langle\{0,1\}, & ^{2}\rangle \\ \langle\{0,1\}, & \{1,2\}\rangle \\ \langle\{0,1\}, & \{1,2\}\rangle \end{pmatrix} \qquad (b)\ \begin{pmatrix} \langle\{0,1\}, & \{1,2\}\rangle \\ \langle\{0,1\}, & \{1,2\}\rangle \\ \langle 0 & , \{1,2\}\rangle \end{pmatrix}$$

We know of no other means to compare the strength of pruning of the two models other than through empirical observation, as presented in Table 3.[1] For a thorough comparison, all solutions are sought. It is infeasible to study all variable orderings, hence columnwise ordering is adopted following previous work on this problem [10]. It is clear from the empirical results that LEX outperforms MS substantially.

n	LEX	MS	LEX′	MS′
3	4	3	4	0
5	29	617	29	22
7	103,671	34,312,243	103,671	63,221
9	130,841,630	>10hrs	130,841,630	107,424,393

Table 3. Solver choices: find all solutions to sports scheduling.

From LEX we can derive, through implication, a new model, LEX′. Since all of the rows of *schedule* must be distinct, the non-strict ordering constraints on the rows are replaced with strict ordering constraints. No other implied constraints are apparent to us.

From MS we can derive, through implication, a new model, MS′. We begin by replacing the non-strict ordering constraints on the rows with strict ones. Next, observe that the $n-1$ rows of *schedule* will be instantiated with the n subsets of cardinality $n-1$ of *teams*. Hence the last team does not play in the first week, so it can be removed from the domain of all variables in the first row. We can continue through all n rows, removing team $(n-(i-1))$ from the domains all variables in the ith row. Now the multiset ordering constraints on the rows are satisfied by all assignments and therefore can be discarded without sacrificing pruning; all the information contained in these ordering constraints has been pushed into the domains. This is another example of model simplification.

Table 3 presents results for the reformulated models. LEX′ does not improve over LEX as the strict ordering constraints only achieve extra pruning when two rows could become equal; the problem constraints forbid this. MS′ improves over MS substantially, further emphasising the need to explore possible inferences after symmetry breaking. Also, though MS was vastly inferior to LEX, MS′ produces a significantly smaller search tree than LEX′. This underlines our second point: a symmetry-breaking scheme should be chosen in the context of the constraints implied by it.

5 Static v. Dynamic Symmetry Breaking

We have so far considered symmetry breaking by adding constraints to the model before search. How might dynamic symmetry-breaking techniques, such as Backofen and Will's technique [1], SBDS [14], or SBDD [4] also support implied constraint generation? These systems dynamically add constraints or examine the search tree to prune areas of the search space symmetric to those already explored. Hence, there are no statically-posted symmetry-breaking constraints from which to make derivations.

Reconsider the 3-fractions puzzle in Section 4.1 and assume that the fractions are taken as the symmetrical objects, but

[1] Times omitted: for reasons that cannot be given briefly, we believe times here reflect solver idiosyncrasies, not model quality.

without the ordering given in (5). Now, since G/HI is not distinguished from the other two fractions, it is not possible to derive an implied constraint as strong as, say, $G \geq 4$. At most, it is possible to say that one of A, D and G must be greater than or equal to 4; this is clearly weaker.

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 [17]. 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 and sport scheduling problems, 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 3 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. 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 the pair belongs to. Occurrence constraints on a_{xy} ensure the correct number of members of each part. This, however, is considerably more expensive to maintain than the decomposition presented in Section 3. Initial experiments with this approach (omitted for space reasons) suggest that the performance is poor.

6 Towards Automation

Our goal here has been to contribute to a systematic understanding of the modelling process, which we ultimately intend to formalise and automate. We aim to formulate formal rules for identifying promising symmetry-breaking constraints and incorporate them into an automated modelling system [8]. Reflecting on this study, we present two informal rules.

The first rule can generate all of the symmetry-breaking constraints and many of the implied constraints considered in the 3-fractions puzzle. If a CSP contains (or entails) a disjunction (say, $C_1 \vee \cdots \vee C_n$) we can, in some cases, introduce a symmetry-breaking constraint so that the resulting CSP entails a particular disjunct, C_i. We can then replace the weak disjunction with the strong constraint, C_i. The rule applies to a CSP that contains (or implies) a constraint of the form

$$t[\vec{X_1}] \leq c \ \vee \ \cdots \ \vee \ t[\vec{X_n}] \leq c \qquad (15)$$

where c is a constant, $t[\vec{X_1}]$ is a term whose variables are $\vec{X_1}$, and each $t[\vec{X_i}]$ is $t[\vec{X_1}]$ where each occurrence of a variable in $\vec{X_1}$ has been replaced with the corresponding variable in $\vec{X_i}$. The vectors $\vec{X_1}, \ldots, \vec{X_n}$ must also be symmetric to each other (i.e. a variable symmetry) in the CSP. Then we can introduce the symmetry-breaking constraint $t[\vec{X_1}] \leq \cdots \leq t[\vec{X_n}]$ and replace constraint (15) with $t[\vec{X_1}] \leq c$.

The second rule is used to introduce *strict* ordering constraints in breaking symmetry. The reader will recognise that this modelling pattern arises frequently. It is capable of generating the strict ordering constraints on the rows in both the LEX′ and MS′ models of the sports scheduling problem. If a CSP has symmetric terms that cannot be "equivalent" under some ordering, then we can impose that ordering strictly among the terms. The justification is that we can break the symmetry by imposing a non-strict ordering on the terms and, since the terms cannot be equivalent, the non-strict ordering can be strengthened to a strict ordering. The rule applies to a CSP that contains a set of terms, $t[\vec{X_1}], \ldots, t[\vec{X_n}]$, such that the vectors $\vec{X_1}, \ldots, \vec{X_n}$ are pairwise symmetric and the CSP entails that no two of the terms are equal. Then we can introduce the constraint $t[\vec{X_1}] < \cdots < t[\vec{X_n}]$.

7 Conclusion

We have identified and documented an important *pattern* in the constraint modelling process: the role of symmetry breaking in deriving powerful implied constraints. We extended this basic pattern to show that, to obtain the best model, the selection of symmetry-breaking constraints must consider the the implied constraints that they enable. Finally, we discussed automating the selection of a symmetry-breaking method, and the consequent derivation of implied constraints, which is the direction our research is now taking.

Acknowledgements Ian Miguel is supported by a UK-Royal Academy of Engineering/EPSRC Post-doctoral Fellowship. We thank Ian Gent, Warwick Harvey, Bernadette Martínez-Hernández, Barbara Smith, and our anonymous referees.

REFERENCES

[1] R. Backofen and S. Will. Excluding symmetries in concurrent constraint programming. *Proc. 5th Int. Conf. on Princ. & Pract. of CP* (LNCS 1713), 73-87, 1999.

[2] J. Crawford, M.L. Ginsberg, E. Luks, A. Roy. Symmetry-breaking predicates for search problems. *Proc. Princ. of Knowledge Representation & Reasoning*, 148-159, 1996.

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

[4] T. Fahle, S. Shamberger, M. Sellman. Symmetry breaking. *Proc. 7th Int. Conf. on Princ. & Pract. of CP* (LNCS 2239), 93-107, 2001.

[5] P. Flener, A.M. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, T. Walsh. Matrix modelling: Exploiting common patterns in constraint programming. *Proc. Int. Wshop on Reformulating CSPs*, 227-41, 2002.

[6] P. Flener, A.M. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, J. Pearson, T. Walsh. Breaking row and column symmetries in matrix models. *Proc. 8th Int. Conf. on Princ. & Pract. of CP* (LNCS 2470), 462-476, 2002.

[7] A.M. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, T. Walsh. Global constraints for lexicographic orderings. *Proc. 8th Int. Conf. on Princ. & Pract. of CP* (LNCS 2470), 93-108, 2002.

[8] A.M. Frisch, C. Jefferson, B. Martínez-Hernández, I. Miguel. *The Rules of Modelling: Automatic Generation of Constraint Programs.* APES Technical Report 85, 2004.

[9] A.M. Frisch, C. Jefferson, I. Miguel. Constraints for breaking more row and column symmetries. *Proc. 9th Int. Conf. on Princ. & Pract. of CP* (LNCS 2833), 318-332, 2003.

[10] A.M. Frisch, I. Miguel, Z. Kiziltan, B. Hnich, T. Walsh. Multiset ordering constraints. *Proc. 18th Int. Joint Conf. on AI*, 2003.

[11] A.M. Frisch, I. Miguel, T. Walsh. Extensions to proof planning for generating implied constraints. *Proc. 9th Symp. on Integration of Symb. Comp. & Mech. Reasoning (Calculemus 01)*, 130-141, 2001.

[12] A.M. Frisch, I. Miguel, T. Walsh. Symmetry and implied constraints in the steel mill slab design problem. *Proc. CP'01 Wshop on Modelling and Problem Formulation*, 8-15, 2001.

[13] I.P. Gent, W. Harvey, T. Kelsey. Groups and constraints: Symmetry breaking during search. *Proc. 8th Int. Conf. on Princ. & Pract. of CP* (LNCS 2470), 415-430, 2002.

[14] I.P. Gent, B.M. Smith. Symmetry breaking during search in constraint programming. *Proc. 14th Euro. Conf. on AI*, 599-603, 2000.

[15] J. Pearson. Comma-free codes. *Proc. 3rd Int. Wshop on Symmetry in Constraint Satisfaction Problems*, 161-167, 2003.

[16] L. Proll, B.M. Smith. ILP and constraint programming approaches to a template design problem. *INFORMS Journal of Computing* 10, 265-275, 1998.

[17] J-F.Puget. Symmetry breaking using stabilizers. *Proc. 9th Int. Conf. on Princ. & Pract. of CP* (LNCS 2833), 585-599, 2003.

[18] M. Sellmann, W. Harvey. Heuristic constraint propagation (using local search for incomplete pruning and domain filtering of redundant constraints for the social golfer problem). *Proc. 4th Int. Wshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimisation Problems (CPAIOR)*, 191-204, 2002.

[19] T. Walsh. Constraint patterns. *Proc. 9th Int. Conf. on Princ. & Pract. of CP* (LNCS 2833), 53-64, 2003.