

# Constraints for Breaking All Row and Column Symmetries in a Three-by-Two Matrix

Alan M. Frisch<sup>1</sup> and Warwick Harvey<sup>2</sup>

<sup>1</sup> Artificial Intelligence Group, Department of Computer Science  
University of York, York, United Kingdom, [frisch@cs.york.ac.uk](mailto:frisch@cs.york.ac.uk)

<sup>2</sup> IC-Parc, Imperial College London, Exhibition Road,  
London SW7 2AZ, United Kingdom, [wh@icparc.ic.ac.uk](mailto:wh@icparc.ic.ac.uk)

**Abstract.** Constraint programs containing a matrix of two (or more) dimensions of decision variables often have row and column symmetries: in any assignment to the variables, the values assigned to any two rows can be swapped and the values assigned to any two columns can be swapped without affecting whether or not the assignment is a solution. This introduces an enormous amount of redundancy when searching a space of partial assignments. It has been shown previously that one can remove some, *but not all*, of these symmetries by extending such a program with constraints that require the rows and columns to be lexicographically ordered. This paper identifies a fully-simplified set of constraints that breaks *all* row and column symmetry in a matrix with three columns and two rows.

## 1 Introduction

A common pattern arising in finite domain constraint programs is the matrix of decision variables with two or more dimensions [3]. In two-dimensional matrices it is often the case that some or all of the rows are interchangeable and some or all of the columns are interchangeable. That is, an assignment to the variables in the matrix is a solution if and only if it is still a solution after swapping the values assigned to two of the interchangeable rows or swapping the values assigned to two of the interchangeable columns. This is called *row and column symmetry*.

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 extra constraints to the model, so-called *symmetry breaking constraints*.

Flener et al [2] studied index symmetry and showed that one can *consistently* add the symmetry-breaking constraint, called *lex*<sup>2</sup>, that both the rows and the columns are lexicographically ordered.<sup>3</sup> This means that for every assignment to the variables that does not satisfy the symmetry-breaking constraint, there is

---

<sup>3</sup> Though working in a different context, Shlyakhter [9] independently showed the consistency.

a symmetric assignment that does. They also showed that for certain problems imposing  $lex^2$  can make the difference between solving and failing to solve the problem. Frisch et al [6] introduced an efficient algorithm for maintaining generalised arc-consistency on the constraint that one vector (a row or column of a matrix) is lexicographically less than another.

Independently, Flener et al [2] and Shlyakhter [9] also showed that  $lex^2$  does not break all row and column symmetries. That is,  $lex^2$  is *incomplete* in that it can be satisfied by two symmetrical assignments. Consider the following two matrices. Both satisfy  $lex^2$ , but the second can be obtained from the first by swapping the rows and rotating the columns to the right.

$$\begin{pmatrix} 2 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 2 \end{pmatrix}$$

In this paper we present, including derivation, a set of lexicographic constraints that is guaranteed to break all row and column symmetries for a  $3 \times 2$  matrix, and that we believe to be minimal (in that no simpler set of lexicographic constraints exists that is complete for domains of arbitrary size).

Whether or not completeness is required depends on the problem to be solved; one example is when one wishes to find all solutions without any symmetric duplicates. In such a case, one either needs to use a complete symmetry breaking approach, or add a separate pass over the solutions to filter out duplicates (e.g. by encoding the solutions as graphs and using a graph isomorphism package such as *nauty* [8] to produce canonical labellings). It is beyond the scope of this paper to compare the two approaches.

## 2 Terminology

We are concerned with finite domain constraint satisfaction problems so every variable is associated with a finite domain of values. An *assignment* maps every variable to a member of its domain.

A set of constraints  $S$  *logically implies* another set of constraints  $S'$  if every assignment that satisfies every member of  $S$  also satisfies every member of  $S'$ . If  $S$  and  $S'$  logically imply each other, then they are said to be *logically equivalent*.

An  $n \times m$  matrix has  $n$  columns and  $m$  rows. We number the columns  $1, \dots, n$  from left to right and the rows  $1, \dots, m$  from top to bottom.

A row of a matrix can be treated as a vector by reading it left to right and a column of a matrix can be treated as a vector by reading it top to bottom. This paper only deals with ordering non-empty vectors of equal size, so we shall simplify the presentation by assuming this throughout the paper. One vector,  $\mathbf{x}$ , is defined to be *lexicographically less than or equal to* another,  $\mathbf{y}$ , (written  $\mathbf{x} \leq_{lex} \mathbf{y}$ ) if  $\mathbf{x} = \mathbf{y}$  or  $x_i < y_i$ , where  $i$  is the smallest index such that  $x_i \neq y_i$ .

## 3 Complete Symmetry Breaking

One way to break all the symmetries in a matrix is to impose the row-wise lex-leader constraints, which are a specific case of the more general lex-leader

constraints introduced by Crawford et al. [1]. The row-wise lex-leader constraint is derived by considering a matrix of distinct variables and all matrices symmetric to it. Each matrix is converted to a string of variables by scanning the matrix row-wise, left-to-right, top-to-bottom. For example, the  $3 \times 2$  matrix of variables

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

yields the string  $ABCDEF$ . From these strings we produce a set of constraints asserting that the string from the original matrix—called the *lex leader*—is lexicographically less than or equal to each of the other strings from the symmetrically equivalent matrices. Thus, continuing our example, there are 11 other matrices that can be produced by permuting the rows and columns of the above matrix. Each of these yields a string of variables that is asserted to be lexicographically greater than or equal to  $ABCDEF$ ; thus 11 constraints, called the row-wise lex leader constraints, are generated:

- (1)  $ABCDEF \leq_{lex} ACBDFE$
- (2)  $ABCDEF \leq_{lex} BCAEFD$
- (3)  $ABCDEF \leq_{lex} BACEDF$
- (4)  $ABCDEF \leq_{lex} CABFDE$
- (5)  $ABCDEF \leq_{lex} CBAFED$
- (6)  $ABCDEF \leq_{lex} DFEACB$
- (7)  $ABCDEF \leq_{lex} EFDBCA$
- (8)  $ABCDEF \leq_{lex} EDFBAC$
- (9)  $ABCDEF \leq_{lex} FDECAB$
- (10)  $ABCDEF \leq_{lex} FEDCBA$
- (11)  $ABCDEF \leq_{lex} DEFABC$

Since the row-wise lex-leader constraints are generated by the method proposed by Crawford et al. [1], we know that they are consistent and complete. It should be noted that this does not generally provide an effective way to break row and column symmetries since for an  $n \times m$  matrix it yields  $n! \cdot m! - 1$  constraints.

## 4 Simplifying the Constraints

We now turn our attention to the task of simplifying the 11 symmetry-breaking constraints. The goal is to produce the simplest possible set of lexicographic ordering constraints that are logically equivalent to the original set.

We present and use two simplification rules. In presenting these rules we use the following meta-symbols.  $X$  and  $Y$  denote arbitrary finite domain variables. Strings of zero or more domain variables are denoted by  $\alpha$ ,  $\beta$ ,  $\gamma$  and  $\delta$ . If any of these Greek letters have superscripts, then the superscript indicates the length of the string.

First consider a general rule for simplifying a single lexicographic ordering constraint.

**Rule 1** *If we have a constraint  $C$  of the form  $\alpha^n X \beta \leq_{lex} \gamma^n Y \delta$  and  $\alpha = \gamma$  logically implies  $X = Y$ , then we may replace  $C$  with  $\alpha \beta \leq_{lex} \gamma \delta$ .*

Essentially, this rule states that if, at the time  $X$  and  $Y$  become significant for this constraint (i.e.  $\alpha = \gamma$ ), we know that they must be equal, then we can leave them out of the constraint because they will have no effect. Note the special case where  $X$  and  $Y$  are the same variable:

**Rule 1'** *If we have a constraint of the form  $\alpha^n X \beta \leq_{lex} \gamma^n X \delta$  then we may replace it with  $\alpha \beta \leq_{lex} \gamma \delta$ .*

**Proposition 1.** *If an application of Rule 1 allows the replacement of one constraint with another, then the two constraints are logically equivalent.*

Using this rule, each of the 11 lex-leader constraints can be simplified to a logically equivalent constraint:

- (1)  $ABCDEF \leq_{lex} ACBDFE \longrightarrow BE \leq_{lex} CF$
- (2)  $ABCDEF \leq_{lex} BCAEFD \longrightarrow ABDE \leq_{lex} BCEF$
- (3)  $ABCDEF \leq_{lex} BACEDF \longrightarrow AD \leq_{lex} BE$
- (4)  $ABCDEF \leq_{lex} CABFDE \longrightarrow ABDE \leq_{lex} CAFD$
- (5)  $ABCDEF \leq_{lex} CBAFED \longrightarrow AD \leq_{lex} CF$
- (6)  $ABCDEF \leq_{lex} DFEACB \longrightarrow ABC \leq_{lex} DFE$
- (7)  $ABCDEF \leq_{lex} EFDBCA \longrightarrow ABCDE \leq_{lex} EFDBC$
- (8)  $ABCDEF \leq_{lex} EDFBAC \longrightarrow ABC \leq_{lex} EDF$
- (9)  $ABCDEF \leq_{lex} FDECAB \longrightarrow ABCDE \leq_{lex} FDECA$
- (10)  $ABCDEF \leq_{lex} FEDCBA \longrightarrow ABC \leq_{lex} FED$
- (11)  $ABCDEF \leq_{lex} DEFABC \longrightarrow ABC \leq_{lex} DEF$

We conjecture that none of these individual constraints can be simplified further. That is, none of the constraints on their own is logically equivalent to a shorter lexicographic ordering constraint. So we now turn our attention to simplifying conjunctions of constraints.

Further simplifications can be performed using the following rule:

**Rule 2** *If we have a set of constraints  $\mathcal{C}$  of the form  $\mathcal{C}' \cup \{\alpha^n \beta \leq_{lex} \gamma^n \delta\}$ , where  $\mathcal{C}'$  is a set of constraints, and  $\mathcal{C}' \cup \{\alpha = \gamma\}$  logically implies  $\beta \leq_{lex} \delta$ , then we may replace  $\mathcal{C}$  with  $\mathcal{C}' \cup \{\alpha \leq_{lex} \gamma\}$ .*

Note that if  $n = 0$  in Rule 2, the lexicographic constraint reduces to a tautology and may be discarded:

**Rule 2'** *If we have a set of constraints  $\mathcal{C}$  of the form  $\mathcal{C}' \cup \{\beta \leq_{lex} \delta\}$ , where  $\mathcal{C}'$  is a set of constraints, and  $\mathcal{C}'$  logically implies  $\beta \leq_{lex} \delta$ , then we may replace  $\mathcal{C}$  with  $\mathcal{C}'$ .*

**Proposition 2.** *If an application of Rule 2 allows the replacement of one set of constraints with another, then the two sets of constraints are logically equivalent.*

Using this rule we obtain the following simplifications, which we will explain.

$$\begin{array}{llll}
(7) & ABCDE \leq_{lex} EFDBC & \longrightarrow & ABCD \leq_{lex} EFDB \\
(2) & ABDE \leq_{lex} BCEF & \longrightarrow & true \\
(4) & ABDE \leq_{lex} CAFD & \longrightarrow & true \\
(9) & ABCDE \leq_{lex} FDECA & \longrightarrow & ABC \leq_{lex} FDE \\
(5) & AD \leq_{lex} CF & \longrightarrow & true
\end{array}$$

The simplification of (7) uses a single application of Rule 2. Suppose that  $ABCD = EFDB$ . In particular,  $A = E$  and  $B = D = C$ . (3) logically implies  $A \leq_{lex} B$ , which in this context is equivalent to  $E \leq_{lex} C$ . Hence by Rule 2,  $ABCDE \leq_{lex} EFDBC$  simplifies to  $ABCD \leq_{lex} EFDB$ .

The simplification of (2) uses four applications of Rule 2. For the first application, suppose that  $ABD = BCE$ . Then  $B = C$ , so (1) logically implies  $E \leq_{lex} F$ . Hence by Rule 2,  $ABDE \leq_{lex} BCEF$  simplifies to  $ABD \leq_{lex} BCE$ . For the second application, suppose that  $AB = BC$ . Then  $A = B$ , so (3) logically implies  $D \leq_{lex} E$ . Hence  $ABD \leq_{lex} BCE$  simplifies to  $AB \leq_{lex} BC$ . For the third application, no supposition is needed. (1) entails  $B \leq_{lex} C$ , so  $AB \leq_{lex} BC$  simplifies to  $A \leq_{lex} B$ . Finally, (3) entails  $A \leq_{lex} B$ , so an application of Rule 2' results in the constraint being reduced to a tautology.

The simplification of (4) also uses four applications of Rule 2. For the first application, suppose  $ABD = CAF$ . Then  $A = B = C$  and  $D = F$ . (1) and  $B = C$  logically implies  $E \leq_{lex} F$ , which implies  $E \leq_{lex} D$  since  $D = F$ . Hence  $ABDE \leq_{lex} CAFD$  simplifies to  $ABD \leq_{lex} CAF$ . For the second application, suppose  $AB = CA$ . Thus  $A = B = C$ , which together with (5) logically implies  $D \leq_{lex} F$ . Hence  $ABD \leq_{lex} CAF$  simplifies to  $AB \leq_{lex} CA$ . For the third application, suppose  $A = C$ . (1) entails  $B \leq_{lex} C$ , which together with  $A = C$  entails  $B \leq_{lex} A$ . Hence  $AB \leq_{lex} CA$  simplifies to  $A \leq_{lex} C$ . Finally, (5) entails  $A \leq_{lex} C$ , so an application of Rule 2' results in the constraint being reduced to a tautology.

The simplification of (9) uses one application of Rule 2. Suppose  $ABC = FDE$ ; then  $B = D$  and  $F = A$ . These two equations together with (1) logically imply  $DE \leq_{lex} CA$ . Hence, by Rule 2,  $ABCDE \leq_{lex} FDECA$  simplifies to  $ABC \leq_{lex} FDE$ .

Finally, the simplification of (5) requires only one rule application. Since the lexicographic ordering is transitive, (3) and (1) imply (5), which thus reduces to a tautology by Rule 2'.

The resulting set of constraints (with the tautologies removed) is:

$$\begin{array}{ll}
(1) & BE \leq_{lex} CF \\
(3) & AD \leq_{lex} BE \\
(6) & ABC \leq_{lex} DFE \\
(7) & ABCD \leq_{lex} EFDB \\
(8) & ABC \leq_{lex} EDF \\
(9) & ABC \leq_{lex} FDE \\
(10) & ABC \leq_{lex} FED \\
(11) & ABC \leq_{lex} DEF
\end{array}$$

Because these constraints were derived from the row-wise lex-leader constraints by equivalence-preserving transformations, we know that these constraints are correct and complete.

**Theorem 3.** *The above constraints are a correct and complete set of symmetry-breaking constraints for the row and column symmetries in a three-by-two matrix.*

Based on the discussion that follows we make a conjecture:

*Conjecture 1.* There is no set of lexicographic ordering constraints that is logically equivalent to and simpler (having fewer or shorter constraints) than the above constraints.

Suppose we were to replace (7) by a slightly weaker constraint:

$$(7') \quad ABC \leq EFD$$

The resulting constraints are quite regular in structure:

- Constraints (1) and (3) constrain the three columns to be in lexicographic order.
- Constraint (11) constrains the two rows to be in lexicographic order.
- The six constraints—(3), (6), (7'), (8), (9) and (10)—constrain the first row to be lexicographically less than or equal to each of the six permutations of the second row.

Unfortunately they are no longer sufficient to break all symmetry: the following two matrices can be obtained from each other by permuting rows and columns,<sup>4</sup> yet both satisfy constraints (1), (3), (6), (7'), (8), (9), (10) and (11).

$$\begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix} \qquad \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix}$$

Only the one on the right satisfies the full-strength (7).

## 5 Past and Future Directions

The results presented here open the door to many future directions. As these results were made public two years ago (though not in writing), some of the future directions are in the past.

In September 2001 we recognised that the lex-leader method of Crawford et al [1] could be used to generate a complete and consistent set of symmetry breaking constraints for row and column symmetries. Realizing that for an  $n \times m$  matrix this method generates a set of  $n! \cdot m! - 1$  constraints, we wondered whether this set could be simplified to a reasonable size. We completed our simplification

---

<sup>4</sup> The second matrix is obtained from first by swapping rows and rotating the columns to the left.

of the constraints for a  $3 \times 2$  matrix in November 2001 and, in presenting a paper at the SymCon'01 Workshop, the first author displayed the simplification [5].

We know of no one in the constraint programming community who previously considered using the lex-leader method for row and column symmetries; indeed, the first proof published in the constraint programming literature of the consistency of the  $lex^2$  constraints does not appeal to the the lex-leader method [2].

However, independently, in June 2001 Shlyakhter published a paper [9] that considered row and column symmetries in the  $n$ -dimensional Boolean matrices that represent  $n$ -ary relations. He showed that for these matrices  $lex^n$  is a consistent symmetry breaking constraint by observing that it is embedded in the row-wise lex-leader constraints.

In 2002, Flener and Pearson [4] reviewed our results on the  $3 \times 2$  matrix and related some of our simplifications to the symmetry group formed by row and column permutations. They also pointed out that further simplifications are possible when the variables in the matrix have a limited domain size.

Our work shows that the row-wise lex-leader constraints for the  $3 \times 2$  matrix entail six lexicographic ordering constraints stipulating that the first row is lexicographically less than or equal to each of the six permutations of the second row. This observation leads one to speculate whether this is a manifestation of a general pattern. Frisch, Miguel and Jefferson [7] investigated this and determined that the row-wise lex-leader constraints for any  $n \times m$  matrix entail  $(m - 1) \cdot n!$  lexicographic constraints stipulating that the first row is lexicographically less than or equal to all permutations of every other row. For a matrix  $M$ , they call the conjunction of these constraints  $allperm(M)$ . Furthermore, they presented an algorithm that enforces generalised arc consistency on  $allperm(M)$  in time  $O(m \cdot (n + d))$ , where  $d$  is the domain size of the variables in  $M$ . For use in cases where  $d$  is large, they provide an alternative implementation whose run time is  $O(m \cdot n \cdot \log n)$ .

These results of Frisch, Miguel and Jefferson answer the open question of whether the row-wise lex-leader constraints can be simplified to a set of lexicographic constraints whose size is polynomial in the size of the matrix. Since  $allperm(M)$  is a conjunction of  $(m - 1) \cdot n!$  constraints, the answer is no—not even close. However, their results point out that the existence of a polynomial-sized set of lexicographic constraints may not be necessary for efficient symmetry breaking: in at least this one case it is possible to efficiently enforce a factorial number of lexicographic constraints.

Whether all row and column symmetries can be broken efficiently remains an open question. A more pragmatic open question is whether there are efficiently-enforceable symmetry-breaking constraints that are non-trivially stronger than the combination of  $lex^2$  and  $allperm$ . Our study of the  $3 \times 2$  matrix provided the vital clue that led to the identification of the  $allperm$  constraint. Beyond this, it is hard to see what clues are provided by the  $3 \times 2$  case. Would a study of symmetry-breaking in the  $3 \times 3$  matrix provide a useful clue?

Another interesting and potentially valuable line of research is the development of an automated method for simplifying a set of lexicographic constraints. The simplification presented in this paper provides examples of the kinds of steps that would need to be automated. Chris Jefferson and Warwick Harvey have both made initial attempts at formulating a simplification algorithm.

We can suggest several possible uses of an automated simplification system. (1) For a given matrix, the system could provide some constraints that could be added usefully to the combination of *lex*<sup>2</sup> and *allperm*. (2) As suggested above, simplification of constraints for matrices larger than  $3 \times 2$  may prove useful in further research. But beyond the  $3 \times 2$  case the manual simplification quickly becomes unmanageable. (3) The study of the row-wise lex-leader constraints has proved fruitful. Perhaps studying the lex-leader constraints produced by orderings other than row-wise would also prove fruitful. Automated simplification would facilitate this. (4) Not only might these other lex-leader constraints be useful for research into symmetry, they might also be effective as symmetry breaking constraints themselves. The amount of search-space pruning obtained by the row-wise lex-leader constraints varies greatly with the variable ordering used by the search. For certain variable orderings it might be much better to use other lex-leader constraints. For this application it would be useful, perhaps necessary, to generate and simplify these constraints automatically.

*Acknowledgements.* This research is supported by UK-EPSRC grant number GR/N16129. We thank Pierre Flener, Brahim Hnich, Chris Jefferson, Zeynep Kiziltan, Ian Miguel, Justin Pearson and Toby Walsh for useful discussions.

## References

1. James Crawford, Matthew L. Ginsberg, Eugene Luks, and Amitabha Roy. Symmetry-breaking predicates for search problems. In Luigia Carlucci Aiello, Jon Doyle, and Stuart Shapiro, editors, *KR'96: Principles of Knowledge Representation and Reasoning*, pages 148–159. Morgan Kaufmann, San Francisco, California, 1996.
2. P. Flener, A.M. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, J. Pearson, and T. Walsh. Breaking row and column symmetries in matrix models. In P. van Hentenryck, editor, *Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming*, pages 462–476, 2002.
3. P. Flener, A.M. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, and T. Walsh. Matrix modelling: Exploiting common patterns in constraint programming. In A.M. Frisch, editor, *Proceedings of the International Workshop on Reformulating Constraint Satisfaction Problems*, pages 27–41, 2002.
4. P. Flener and J. Pearson. Breaking all the symmetries in matrix models: Results, conjectures, and directions. In P. Flener and J. Pearson, editors, *Proceedings of SymCon'02: The Second International Workshop on Symmetry in Constraint Satisfaction Problems*, 2002.
5. Alan M. Frisch, Ian Miguel, Toby Walsh, Pierre Flener, Brahim Hnich, Zeynep Kiziltan, and Justin Pearson. Reducing symmetry in matrix models. Talk presented at the First International Workshop on Symmetry in Constraint Satisfaction Problems, December 2001. Slides at [www.cs.york.ac.uk/aig/projects/IMPLIED/docs/SymMxCP01.ppt](http://www.cs.york.ac.uk/aig/projects/IMPLIED/docs/SymMxCP01.ppt).

6. A.M. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, and T. Walsh. Global constraints for lexicographic orderings. In P. van Hentenryck, editor, *Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming*, pages 93–108, 2002.
7. A.M. Frisch, C. Jefferson, and I. Miguel. Constraints for breaking more row and column symmetries. In F. Rossi, editor, *Proceedings of the Ninth International Conference on Principles and Practice of Constraint Programming*, 2003.
8. B. D. McKay. *nauty* user's guide (version 1.5). Technical Report TR-CS-90-02, Computer Science Department, Australian National University, 1990.
9. Ilya Shlyakhter. Generating effective symmetry-breaking predicates for search problems. In Henry Kautz and Bart Selman, editors, *Electronic Notes in Discrete Mathematics*, volume 9. Elsevier Science Publishers, 2001.