

# Confluence of reduction rules for lexicographic ordering constraints

Andrew Grayland<sup>1</sup>, Ian Miguel<sup>1</sup> and Colva M. Roney-Dougal<sup>2</sup>  
(1. School of Comp. Sci., 2. School of Maths and Stats), St Andrews, UK.  
(andyg@cs, ianm@cs, colva@mcs).st-and.ac.uk

## Abstract

The lex leader method for breaking symmetry in a constraint model typically produces a large set of lexicographic ordering constraints. Several rules have been proposed in the literature to reduce such a set of constraints whilst preserving logical equivalence. These reduction rules are not in general confluent: they may reach more than one fixpoint, depending on the order of application. We characterise the systems of constraints for which the reduction rules are confluent in terms of a simple feature of the input lex constraints named ‘blocks’ and provide preliminary experimental evidence to support the theory that smaller sets of lex constraints lead to faster solve times.

## 1 Introduction

Constraint models often contain *symmetries* that partition the set of assignments into equivalence classes. Symmetries can be exploited by restricting the search for a solution to one member of each class (*symmetry breaking*), dramatically reducing search. A commonly-used symmetry-breaking technique adds constraints to the model that break symmetries statically. Crawford *et al.*'s *lex leader* method [Crawford *et al.*, 1996] adds a lexicographic ordering constraint (hereafter, *lex constraint*) per symmetry to allow only one element of each equivalence class.

The drawback of this approach is that the number of lex constraints required can become very large. Symmetries can be captured using the mathematical formalism of a group. A group on  $n$  points can have up to  $n!$  elements, requiring  $n!$  lex constraints. Puget [Puget, 2005] has identified a special case, where each variable must be assigned a distinct value, in which the set of ordering constraints collapses to just  $n - 1$  binary inequalities. Frisch and Harvey [Frisch *et al.*, 2003], Öhrman [Öhrman, 2005] and Grayland *et al.* [Grayland *et al.*, 2008] define rules for the reduction of a set of lex constraints to a smaller (in both arity and number), but logically equivalent, set without the need for the `alldifferent` constraint.

The advantage of this method is that we consider the reductions applied to any group and the resulting constraints are a new set of lex constraints. Luks and Roy [Luks *et al.*, 2004] detail a different approach in which the lex constraints are described as Boolean formulae where redundant clauses are pruned, this is extended by Roy in [Roy, 2007]. Here the groups considered are initially bounded to have orbits of size 2, and extensionally to any orbits with some maximum size  $c$  (a fixed constant). Since the resultant constraints are not lex constraints and an efficient propagation algorithm has not yet been discussed we propose our method as an alternative.

This paper shows that there exist *nonconfluent* systems of lex constraints even when the symmetry group is *transitive*. Hence, the final set can vary according to the order of application of the rules. We characterise the systems of lex constraints for which the reduction rules are *confluent* in terms of a simple feature of the input lex constraints named ‘blocks’. Using this characterisation it is possible to determine a minimum reachable fixpoint for some commonly occurring infinite families of groups.

## 2 Background

Frisch and Harvey [Frisch *et al.*, 2003] introduced Rules 1 and 2 to reduce the number and arity of constraints whilst maintaining logical equivalence. If  $\alpha, \beta, \gamma$ , and  $\delta$  are strings of variables, and  $x$  and  $y$  are individual variables, Rules 1 and 2 are stated:

- 1 If we have a constraint  $c$  of the form  $\alpha x \beta \leq_{\text{lex}} \gamma y \delta$ , and  $\alpha = \gamma$  entails  $x = y$  then we may replace it with  $\alpha \beta \leq_{\text{lex}} \gamma \delta$ .
- 2 If  $C$  is a set of constraints of the form  $C' \cup \{\alpha x \leq_{\text{lex}} \gamma y\}$ , and  $C' \cup \{\alpha = \gamma\}$  entails  $x \leq y$ , then replace  $C$  with  $C' \cup \{\alpha \leq_{\text{lex}} \gamma\}$ .

Öhrman [Öhrman, 2005] defines a further Rule 3 which supercedes and is stronger than Rules 1 and 2. Rule 3 extends the stated Rules 1 and 2 in that it allows both the consideration of all pairs of variables in any one lex constraint, provided by Rule 1, and the implications derived from considering the entire set of lex constraints, provided by Rule 2. Unfortunately the support required for removal of the least significant pair remains essentially different from that required for the removal of any

other pair. Whilst we can remove any least significant pair in a lex constraint by showing that it is always less than or equal at the time it is considered, we must show that any other pair is equal at the time it is considered in order to remove it. For this reason we find it useful to remove the action of Öhrman’s Rule 3 that covers the actions of Rule 2 and to restate it as Rule 3’.

3’ If  $C$  is a set of constraints of the form  $C' \cup \{\alpha x \beta \leq_{\text{lex}} \gamma y \delta\}$ , and  $C' \cup \{\alpha = \gamma\}$  entails  $x = y$ , then replace  $C$  with  $C' \cup \{\alpha \beta \leq_{\text{lex}} \gamma \delta\}$ .

It was demonstrated in [Grayland *et al.*, 2008] that in general large sets of lex constraints require a longer solve time than equivalent smaller sets therefore it is desirable to reduce a large set as far as possible. To do so, the reduction rules are applied until a fixpoint is reached.

**Definition 1** A reachable fixpoint for a set  $C$  of lex constraints is a set of lex constraints produced by removing pairs from  $C$  using Rules 2 and 3’ such that no further applications of Rules 2 and 3’ are possible.

However, the reduction rules are not, in general, confluent [Grayland *et al.*, 2008].

**Definition 2** A terminating rewriting system is confluent if the application of the rewrite rules reaches the same fixpoint irrespective of the order in which they are applied.

The proof of nonconfluence in [Grayland *et al.*, 2008] relied on a set of lex constraints derived from an *intransitive* group of symmetries.

**Definition 3** A group of permutations is transitive if every point that the group acts on can be mapped to any other.

It was shown in [Grayland *et al.*, 2008] that Rule 1 is confluent. For the remainder of the paper we utilise Rule 1 as a pre-processing step to further reduction. On the remaining constraints we will use Rule 2 and 3’ for reduction.

For example, consider a constraint  $c: x_1 x_2 \leq_{\text{lex}} x_2 x_1$ . To ensure that  $c$  is satisfied we need only compare a pair of variables if each pair of more significant variables are equal. If  $x_1 = x_2$  then trivially the second pair *must* be equal. Therefore, by Rule 3’ we need only consider the first pair of variables, reducing  $c$  to  $x_1 \leq x_2$  without modifying the set of solutions.

Every permutation can be written as a composition of disjoint cycles. Consider the following permutation in list notation:

$$[C, A, F, D, G, B, H, E]$$

We see that  $f(A) = C$ ,  $f(C) = F$ ,  $f(F) = B$  and  $f(B) = A$ ; similarly,  $f(E) = G$ ,  $f(G) = H$  and  $f(H) = E$ , while  $f(D) = D$ . In cycle notation we generally omit mappings from one point to itself, so we have:

$$(ACFB)(EGH)$$

### 3 Confluence in Lex Leader Reduction

Rules 2 and 3’ are not confluent when applied to arbitrary sets of lex constraints, or to sets of constraints produced by groups that cannot map any variable to any other [Grayland *et al.*, 2008]. We now show that the same holds even for transitive groups.

**Theorem 1** There exists a set of lex constraints for a transitive group for which Rules 2 and 3’ are not confluent.

**Proof** Let  $G$  be `TransitiveGroup(6,6)` in GAP’s library [Conway *et al.*, 1998; The GAP Group, 2007], permuting variables  $A, B, C, D, E, F$ . In cycle notation, the group elements are:

$$\begin{aligned} &(), (CF), (AEC)(BFD), (AECDBF), \\ &(ACBDFE), (BE), (BE)(CF), (AEFDBC), \\ &(AEF)(BCD), (ACEDFB), (ACB)(DFE), (AD), \\ &(AD)(CF), (ABFDEC), (ABF)(CDE), (AFBDCE), \\ &(AFE)(BDC), (AD)(BE), (AD)(BE)(CF), \\ &(ABC)(DEF), (ABCDEF), (AFB)(CED), \\ &(AFEDCB), (ACE)(BDF) \end{aligned}$$

The group contains 24 elements, but using Rules 2 and 3’ we reduce the full set of lex constraints for  $G$  to:

$$\begin{aligned} (1) \quad &ABD \leq_{\text{lex}} BCE, \quad (2) \quad C \leq_{\text{lex}} F, \\ (3) \quad &B \leq_{\text{lex}} E, \quad (4) \quad A \leq_{\text{lex}} D, \\ (5) \quad &ABDE \leq_{\text{lex}} CAFD, \quad (6) \quad AB \leq_{\text{lex}} CA \end{aligned}$$

There are now two pairs for removal, namely pairs  $B \leq A$  in (5) and (6).

**Method 1:** From (5) by Rule 3’, Figure 1. Assume  $A = C$ , then (6) gives  $B \leq A$ . Since (1) states  $A \leq B$ , we deduce  $A = B$ , and remove  $B \leq A$  from (5). We then reach a fixpoint (1), (2), (3), (4), (6),  $ADE \leq_{\text{lex}} CFD$ .

**Method 2:** From (6) by Rule 2, Figure 2. Assume  $A = C$ , then (5) gives  $B \leq A$ , and the second pair of (6) can be removed. After this (6) is now equal to the first pair of (5), and so we remove all of (6) by Rule 2, giving a new fixpoint, (1), (2), (3), (4), (5).  $\square$

It was previously known that Rule 3’ could remove pairs of variables that Rule 2 cannot remove [Öhrman, 2005], but only when applied to sets of lex constraints that are not reduced from the full set for some group.

**Corollary 1** There exists a (transitive) group whose full set of lex constraints contains variables which can be removed by Rule 3’ but not Rule 2.

### 4 Activation Graphs and Confluence

We now characterise the features of lex constraints that make Rules 2 and 3’ non-confluent. In the set of lex constraints in the proof of Theorem 1, non confluence arises because the pair  $B \leq A$  occurs in two constraints. Each copy of this pair can remove the other, but nothing can remove both of them. In this section we prove that this is essentially the only way in which sets of lex constraints can be non confluent under the action of Rules 2 and 3.

Before characterising confluence, we need a precise understanding of which pairs of which constraints are used by Rules 2 and 3' to remove a pair. This precision is achieved by introducing the notion of an *activation graph*. For the remainder of this paper, let  $\mathcal{C} = \{c_1, \dots, c_k\}$  be a set of lex constraints, not necessarily derived from a group. The pair of variables in position  $j$  of constraint  $i$  is denoted  $c_{ij}$ .

**Definition 4** A goal  $c_{\alpha\beta}$  is a pair under consideration for removal by Rules 2 and 3'. Each goal defines an activation graph. An **activation graph**  $G_{\alpha\beta}$  is a digraph which is generated from the assumptions made to remove the goal  $c_{\alpha\beta}$  by Rule 2 or 3'.

The nodes of  $G_{\alpha\beta}$  are  $\{c_{ij} \mid c_i \in \mathcal{C}, j \leq \beta \text{ if } i = \alpha\}$ , and are arranged in  $k$  rows, one for each constraint. The nodes are also arranged in columns from most significant to least significant. The first row is  $c_{\alpha\alpha}$ , truncated immediately after  $c_{\alpha\beta}$ . The order of the other rows does not matter, and they should be considered as a set.

Non-goal nodes can be active or inactive. A node  $c_{ij}$  is active if  $i = \alpha$  and  $j < \beta$ , or  $j = 1$ , or there is an edge from  $c_{i(j-1)}$  to  $c_{ij}$ . An active node  $c_{ij}$  is green if the pair of variables are equal, and is amber otherwise. Inactive nodes are red. The colour of a node may change from red to amber to green, as edges are added, but not in the other direction. When initially constructing the graph the active nodes are  $c_{i1}$  for  $i \neq \alpha$ , which are amber, and  $c_{\alpha j}$  for  $j < \beta$ , which are green.

Let  $c_{ij} \neq c_{\alpha\beta}$ , and assume that  $c_{i(j-1)}$  is active. A justification set for  $c_{ij}$  is a set of nodes  $A_{ij} = \{c_{st} \mid (c_{st}, c_{ij}) \in E(G_{\alpha\beta})\}$ . If  $A_{ij}$  is nonempty then it entails the equality of  $c_{i(j-1)}$ . The justification set  $A_{ij}$  is minimal if for all  $x \in A_{ij}$  the set  $A_{ij} - x$  is insufficient to activate  $c_{ij}$ .

There are several types of edges in  $G_{\alpha\beta}$ :

1. If  $c_{ij} \neq c_{\alpha\beta}$  then there is a directed edge from an active node  $c_{st}$  to  $c_{ij}$  whenever there exists a minimal justification set for  $c_{ij}$  containing  $c_{st}$ . If so, there is an edge from  $c_{i(j-1)}$  to  $c_{ij}$ ,  $c_{i(j-1)}$  is green, and  $c_{ij}$  is amber or green.
2. There is a directed edge from an active node  $c_{ij}$  to  $c_{\alpha\beta}$  when  $c_{ij}$  is a member of a minimal set of nodes entailing  $c_{\alpha\beta}$  (Rule 2) or entailing equality in the variables in  $c_{\alpha\beta}$  (Rule 3').
3. There are directed edges between consecutive nodes in row 1, from more significant to less significant, up to but not including  $c_{\alpha\beta}$ .

The node  $c_{\alpha\beta}$  can be removed by Rules 2 and 3' if and only if there is a directed edge into  $c_{\alpha\beta}$  in  $G_{\alpha\beta}$ .

**Definition 5** An **activation chain** for  $c_{\alpha\beta}$  is a subgraph of  $G_{\alpha\beta}$  whose nodes include  $c_{\alpha\beta}$  and which contains a minimum justification set for each of its nodes. Note that  $c_{\alpha\beta}$  can have more than one activation chain.

Intuitively, an activation chain represents one argument for the removal of  $c_{\alpha\beta}$ . To clarify these concepts, we present activation graphs  $G_{52}$  and  $G_{62}$  for the set

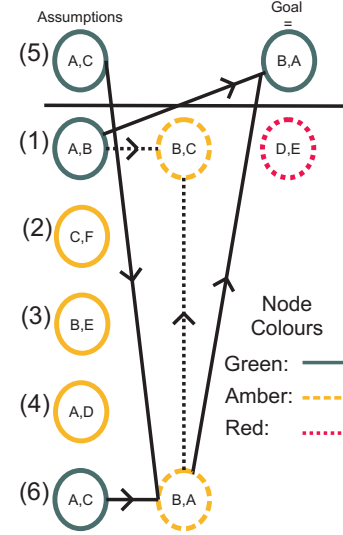


Figure 1: An activation graph for the reduction described in Method 1 of Theorem 1. The solid black edges represent a possible activation chain. Green nodes are equal, amber nodes are less than or equal and red nodes are not active. The goal is  $c_{52}$ ,  $B \leq A$  and the assumption is  $A = C$ .

$\{(1), \dots, (6)\}$  of constraints given in the proof of Theorem 1.

We now show that if an activation chain for  $c_{ij}$  contains  $c_{st}$ , and  $c_{st}$  is removed, then unless all activation chains for  $c_{st}$  include  $c_{ij}$  we can still remove  $c_{ij}$ .

**Lemma 1** Let  $c_{st}$  have at least one activation chain. If there exists an activation chain  $C_1$  for  $c_{ij}$  such that  $c_{st} \notin C_1$  then there exists an activation chain  $C_2$  for  $c_{st}$  such that  $c_{ij} \notin C_2$ . Thus the order of removal  $c_{ij}$  and  $c_{st}$  does not alter the set of reachable fixpoints.

**Proof** If no activation chain for  $c_{st}$  includes  $c_{ij}$  then the result follows. So let  $C_3$  be an activation chain for  $c_{st}$  with  $c_{ij} \in C_3$ .

To construct  $G_{ij}$  we make  $c_{il}$  green for  $l < j$ , and  $c_{m1}$  amber for  $m \neq i$ . All other edges in  $C_1$  are entailed by these initial assumptions. Let  $A_{ij}$  be the set of nodes with directed edges to  $c_{ij}$  in  $C_1$ .

Since  $c_{ij}$  is in  $C_3$  it is active, so  $c_{il}$  is green for  $l < j$  in  $C_3$ . Clearly  $c_{m1}$  is amber or green for  $m \neq i$ , thus all entailments in  $G_{ij}$  are in  $G_{st}$ . In particular, nodes  $A_{ij}$  are the same colour in  $G_{st}$  as in  $C_1$ .

We modify  $C_3$  as follows: first remove  $c_{ij}$ , then add all edges from  $G_{st}$  required to activate all of  $A_{ij}$ , then place a directed edge from every node in  $A_{ij}$  to every node that had an edge from  $c_{ij}$ . Finally, reduce  $C_3$  back to an activation chain. Since  $A_{ij}$  entails  $c_{ij}$ , all active nodes and the goal have a justification set, but  $c_{ij} \notin C_3$ .  $\square$

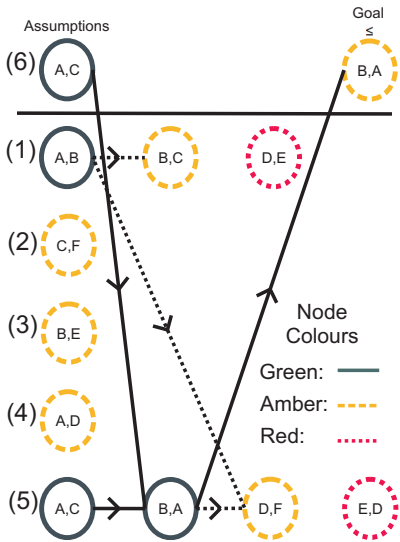


Figure 2: An activation graph for the reduction described in Method 2 of Theorem 1. The solid black edges represent a possible activation chain. Green nodes are equal, amber nodes are less than or equal and red nodes are not active. The goal is  $c_{62}$ ,  $B \leq A$  and the assumption is  $A = C$ .

We now define the generalisation of the two pairs  $B \leq A$  in the proof of Theorem 1 that prevented confluence.

**Definition 6** A block  $B$  is a set of at least two constraint pairs  $c_{ij}$ , where not all of the constraints  $c_i$  have the same fixpoint if reduced first, such that for all  $p \neq q \in B$  the node  $q$  is in at least one activation chain for  $p$ , and conversely all activation chains for  $p$  contain at least one  $q \in B$ .

Our main result shows that a system of lex constraints is confluent reduced by Rules 2 and 3' if and only if it contains no blocks.

**Theorem 2** Let  $\mathcal{C}$  be a set of lex constraints. The reduction of  $\mathcal{C}$  by Rules 2 and 3' is confluent if and only if there is no block in  $\mathcal{C}$ .

**Proof** First we show that a block  $B$  prevents confluence. Since every element of  $B$  has at least one activation chain, any element of  $B$  can be removed. However, not all elements of  $B$  can be removed as the final one will have no remaining activation chains – all such chains contain at least one member of  $B$ , and hence will have been broken. Since more than one fixpoint is possible amongst the constraints containing the pairs in the block,  $\mathcal{C}$  has more than one fixpoint under Rules 2 and 3'.

Now we give a sketch proof that if  $\mathcal{C}$  has no blocks then its reduction by Rules 2 and 3' is confluent. There are two cases. If no activation chain for any element of  $\mathcal{C}$  that can be removed includes any other element that can be removed, then clearly the reduction of  $\mathcal{C}$  is confluent. So suppose that  $c_{ij}$  has an activation chain

including  $c_{st}$ , and that  $c_{st}$  can also be removed. If  $c_{st}$  has an activation chain  $\mathcal{C}$  with  $c_{ij} \notin \mathcal{C}$  then  $c_{ij}$  has activation chains that do not use  $c_{st}$ , by Lemma 1. Hence the order of removing  $c_{ij}$  and  $c_{st}$  does not matter. If instead all activation chains for  $c_{st}$  include the node  $c_{ij}$  then again by Lemma 1 all activation chains for  $c_{ij}$  must include  $c_{st}$ . Hence  $c_{st}$  and  $c_{ij}$  form a block, a contradiction.  $\square$

## 5 Confluence for Some Families of Variable Symmetries

In [Grayland *et al.*, 2008] a number of general formulae for the construction of minimal lex constraints for problems with particular symmetries were defined. We now show that these general formulae produce minimum sets of lex constraints with respect to Rules 2 and 3' and therefore the actions of the Rules is confluent.

**Theorem 3** Let  $P$  be a CSP with  $n$  decision variables  $\{x_1, \dots, x_n\}$  whose symmetry group is the full symmetric group  $S_n$  on variables. Given a lex leader in which the variables are ordered in significance from  $x_1$  to  $x_n$ , Rules 2 and 3' reduce the corresponding lexicographic constraints confluent to:

$$\mathcal{M}_s = \{x_i \leq x_{i+1} : 1 \leq i \leq n-1\}$$

**Proof** We now show that there is only one reachable fix-point.

Note that each pair in  $\mathcal{M}_s$  comes from the most significant non-trivial pair of one or more lex constraints from the full set,  $\mathcal{C}$ . For example, the pair  $x_1 \leq x_2$  occurs as the most significant pair of exactly  $(n-1)!$  lex constraints in  $\mathcal{C}$ . Whilst the pair  $x_2 \leq x_3$  never occurs as the most significant pair, it does occur as the most significant nontrivial pair when the most significant pair is  $x_1 \leq x_1$ . Rule 1 removes all such trivial pairs, and it is shown in [Grayland *et al.*, 2008] that Rule 1 is confluent. We define a new set of lex constraints  $\mathcal{C}_1$  which is  $\mathcal{C}$  after application of Rule 1 to a fix-point.

Recall that the  $j$ th most significant pair in a constraint  $i$  is referred to as  $c_{ij}$ . We now consider which pairs would need to appear on an activation chain in order to remove the pair  $c_{i1}$ , where  $c \in \mathcal{M}_s$ . Since  $c_{i1}$  is the only, and hence last, pair of its constraint we consider reduction by Rule 2. We could use Rule 3', but since Rule 3' requires equality in  $c_{i1}$ , a lack of reduction by Rule 2 implies no possible reduction by 3'. There are no more significant pairs to consider equal. Since there are no assumptions, only the most significant pairs of the other constraints  $\mathcal{C}'_1$ , where  $\mathcal{C}'_1 = \mathcal{C}_1 \setminus c_i$ , may become active in any activation graph with goal  $c_{i1}$ , otherwise we would have some pairs defined as always equal by the lex constraints. Recall that  $c_{i1}$  is of the form  $x_i \leq x_{i+1}$ . Given no assumed equalities the only methods to activate  $c_{i1}$  in an activation chain are to either activate another node  $x_i \leq x_{i+1}$ , or to activate a set of nodes such that  $x_i \leq x_j \leq \dots \leq x_k \leq x_{i+1}$ .

Given the canonical ordering chosen the left hand side of each constraint in  $\mathcal{C}$  is  $x_1x_2\dots x_n$ . The first moved point in a permutation must be mapped to a higher moved point, so, having applied Rule 1 to obtain  $\mathcal{C}_1$ , if  $x_i \leq x_j$  is the first pair of a constraint in  $\mathcal{C}_1$  then  $i < j$ .

Assume there exist more than one active nodes in an activation chain for  $c_{i1}$ . The chain must contain the node  $x_i \leq x_b$  as well as the node  $x_c \leq x_{i+1}$ . Since  $x_i \leq x_j$  is not a possible active node, if  $i > j$  we conclude that  $i < b \leq n$ . Similarly,  $1 \leq c < (i + 1)$  and hence  $b > c$ . Without loss of generality we may assume that if a chain of active nodes exists such that  $x_b \leq \dots \leq x_c$  we may consider there to be an active node  $x_b \leq x_c$ . Since  $b > c$  we know this chain of nodes cannot exist, a contradiction.

Hence, the only remaining method to remove  $c_{i1}$  by Rule 2 is to find a node  $x_i \leq x_{i+1}$  in another constraint  $c_m \in \mathcal{C}_1$  which is active in an activation graph for  $c_{i1}$ . Again, there are no assumptions made so  $x_i \leq x_{i+1}$  must be the most significant pair in  $c_m$ . If such a constraint  $c_m$  exists then we can remove  $c_{i1}$ , however the constraint  $c_m$  has most significant pair  $x_i \leq x_{i+1}$ , and as such, the above proof can be used to define the removal of it. We can continually prune the pair  $c_{i1}$  until no such constraint  $c_m$  exists.

There will always exist the node  $x_i \leq x_{i+1}$  in some constraint, since the last instance of  $x_i \leq x_{i+1}$  will have no active nodes in its activation graph.

It follows that  $\mathcal{M}_s$  is the unique fixpoint and hence the system is confluent.  $\square$

**Definition 7** Let  $P$  be a CSP with  $n$  decision variables,  $\{x_1, x_2, \dots, x_n\}$ . If the symmetry group of  $P$  is a cyclic group of variable symmetries then a complete set  $\mathcal{M}_c$  of symmetry breaking constraints is [Grayland et al., 2008]:

$$\begin{array}{llll}
i = 1 & & x_1 & \leq & x_2 \\
i = 2 & & x_1x_2 & \leq_{\text{lex}} & x_3x_4 \\
i = 3 & & x_1x_2x_3 & \leq_{\text{lex}} & x_4x_5x_6 \\
\vdots & & \vdots & & \vdots \\
n \text{ even} & & x_1x_2 \dots x_{n/2+1} & \leq_{\text{lex}} & x_{n/2+2} \dots x_nx_1x_2 \\
n \text{ odd} & & x_1x_2 \dots x_{(n+1)/2} & \leq_{\text{lex}} & x_{(n+3)/2} \dots x_nx_1 \\
\vdots & & \vdots & & \vdots \\
i = n - 1 & & x_1x_2 \dots x_{n-1} & \leq_{\text{lex}} & x_nx_1 \dots x_{n-2}
\end{array}$$

**Lemma 2** Given a minimal set of lex constraints  $\mathcal{M}_c$  required to break a cyclic group symmetry, if  $i > n/2$  then the assumption of equality in the first  $(i - 1)$  pairs of variables in  $c_i$  produces  $n - i$  distinct equality classes of size greater than 1 between the variables involved in that constraint. If  $i \leq n/2$  then  $(i - 1)$  equality classes of size greater than 1 are produced.

**Proof** We first consider the case that  $i \leq n/2$ . Then  $c_i \in \mathcal{M}_c$  is  $x_1 \dots x_i \leq_{\text{lex}} x_{i+1} \dots x_{2i}$ . All variables in  $c_i$  are distinct, hence  $i - 1$  equality classes of size greater than 1 are created.

For the rest of the proof we assume that  $i > n/2$ , so  $c_i$  is

$$x_1 \dots x_i \leq_{\text{lex}} x_{i+1} \dots x_nx_1 \dots x_{2i-n}.$$

The  $n - i$  most significant pairs in  $c_i$  contain distinct variables  $x_1, \dots, x_{n-i}, x_{i+1}, \dots, x_n$ .

We assume that  $x_1 = x_{i+1}$ ,  $x_2 = x_{i+2}$ ,  $\dots$ ,  $x_{n-i} = x_n$ . After these  $n - i$  pairs of variables there is a repeat of  $x_1x_2x_3 \dots x_{(n-i)}$ , this time on the right hand side. We assume that  $x_1x_2x_3 \dots x_{n-i} = x_{(n-i)+1}x_{(n-i)+2} \dots x_{2(n-i)}$ .

We now have  $n - i$  equality classes, each involving 3 variables, e.g.  $x_1 = x_{i+1} = x_{(n-i)+1}$ . The pattern then continues with  $x_{(n-i)+1}x_{(n-i)+2} \dots x_{2(n-i)}$  appearing on the right hand side, assumed to be equal to the next  $n - i$  variables,  $x_{2(n-i)+1}, \dots, x_{3(n-i)}$ . This pattern continues until the pair of variables under consideration, namely  $x_i \leq x_{2i-n}$ .

As the number of variables in each equality class grows, the additions are variables that have not appeared before. Therefore the initial  $n - i$  equality classes will never merge.

The new variables are always assumed to be equal to a variable currently in an equality class of size greater than 1, so there will never be more than  $n - i$  equality classes of size greater than 1.  $\square$

**Theorem 4** Let  $P$  be a CSP with  $n$  decision variables  $\{x_1, \dots, x_n\}$  whose symmetry group is the cyclic group  $C_n$  on variables. Given a lex leader in which the variables are ordered in significance from  $x_1$  to  $x_n$ , Rules 2 and 3' reduce the corresponding lexicographic constraints confluent to the set  $\mathcal{M}_c$ .

**Proof** We now show that there is only one reachable fixpoint.

Recall that the complete set of non-identity lex constraints  $\mathcal{C}$  generated for a cyclic group symmetry is of the form:

$$\begin{array}{ll}
(c_1) & x_1x_2 \dots x_n \leq_{\text{lex}} x_2x_3 \dots x_nx_1 \\
(c_2) & x_1x_2 \dots x_n \leq_{\text{lex}} x_3x_4 \dots x_nx_1x_2 \\
& \vdots \\
(c_{n-1}) & x_1x_2 \dots x_n \leq_{\text{lex}} x_nx_1 \dots x_{n-1}
\end{array}$$

We begin by showing that it is not possible to remove the least significant pair  $c_{ii}$  in the constraint  $c_i \in \mathcal{M}_c$  by Rule 2 or any pair by Rule 3' even when using the set  $\mathcal{C}' = \mathcal{C} \setminus c_i$  for support, and hence any pair in  $\mathcal{M}_c$  must be in any minimum set.

**RULE 2:** The pair under consideration for removal from  $c_i \in \mathcal{M}_c$  by Rule 2 is  $c_{ii}$ :  $x_i \leq_{\text{lex}} x_j$ , where  $j = 2i$  if  $2i \leq n$  and  $j = 2i - n$  if  $2i > n$ . To prune  $c_{ii}$  we assume  $x_1 \dots x_{i-1} = x_{i+1} \dots x_{2i}$  if  $2i \leq n$  or  $x_1 \dots x_{i-1} = x_{i+1} \dots x_nx_1 \dots x_{2i-n}$  if  $2i > n$ . A constraint  $c_a \in \mathcal{C}'$ , where  $a < i$ , is of lower arity than  $c_i$ . A constraint  $c_b \in \mathcal{C}'$ , where  $b < i$ , is of higher arity than  $c_i$ .

We begin by showing that we cannot activate any pair with  $x_i$  on the left hand side and that  $x_i$  is not equal

to any other variable, therefore we cannot imply that  $x_i \leq x_j$ . This is split into three sections. First we discuss the case where  $2i \leq n$ . Next we look at the constraints  $c_b$  for  $b > i$  where  $2i \leq n$ . Finally we look at the constraints  $c_a$  for  $a < i$  where  $2i \leq n$ . In turn we show that the variable  $x_i$  is never activated on the left hand side of any pair.

First note that if  $2i \leq n$  then there are equality classes  $x_1 = x_{i+1}, x_2 = x_{i+2}, \dots, x_{i-1} = x_{2i-1}$ . The constraints  $c_a$  for  $a < i$  have most significant pairs  $x_1 \leq x_2, x_1 \leq x_3, \dots, x_1 \leq x_{i-1}$ . The constraints  $c_b$  for  $b > i$ , have most significant pairs  $x_1 \leq x_{i+2}, x_1 \leq x_{i+3}, \dots, x_1 \leq x_{n-1}$ . In order to use less significant pairs of variables in these constraints we must show equality in these initial pairs, however they all lie in distinct equality classes. Since we never assume  $x_i$  to be less than or equal to anything  $c_{ii}$  cannot be reduced.

We now assume that  $2i > n$ , so that the pair under consideration for removal from  $c_i \in \mathcal{M}_c$  by Rule 2 is  $x_i \leq x_j$ , where  $j = 2i - n$ .

We now look to see if we can find the variable  $x_i$  on the left hand side of a pair in the constraints  $c_b$  for  $b > i$ . By Lemma 2, the first  $n - i$  decision variables are never assumed to be equal to each other. Notice also that the these  $n - i$  variables are equal to the last  $n - i$  variables respectively, i.e.  $x_1 = x_{i+1}, x_2 = x_{i+2}, \dots, x_{n-i} = x_n$ .

The equality classes not containing  $x_1$  contain the variables  $x_{i+2}, \dots, x_n$ . These are the right hand variables of the most significant pairs of  $c_b$ . Therefore the Rule 2 assumptions of equality alone are not enough to imply equality in these most significant pairs and as such all less significant pairs are not active on an activation graph with goal  $c_{ii}$ .

We now consider the lower arity constraints  $c_a$  for  $a < i$ . There is equality in the most significant pairs of  $c_{n-i}, c_{2(n-i)}, \dots$ . This is simply because the left hand element in the pair is always  $x_1$ , and Lemma 2 shows us that  $x_1$  is equal to every  $(n - i)^{th}$  element. In these constraints, we find that the pairs of variables are matched to those in the equality classes that we are assuming exist from  $c_i$ . This is because the equality relations step along in groups of  $n - i$ .

The first distinct pair of variables in each constraint which are not assumed to be equal is the pair with  $x_i$  on its right hand side, hence at most we may deduce that  $x_i$  is greater than another variable. This new inequality does not imply equality in the most significant pair of any other constraint, and as such we still cannot utilise pairs within them. Since  $x_i$  has not been assumed to be equal to anything else and since nothing we have done so far has implied it to be equal to anything else, we cannot deduce that  $x_i \leq x_j$ .

**RULE 3'**: We now consider application of Rule 3' on the pair  $c_{ik} = x_k \leq x_{k+i}$  where  $i \leq (n/2)$  or  $c_{ik} = x_k \leq x_{k-(n-i)}$  where  $i > (n/2)$ , for  $k < i$ , which is not least significant in the constraint  $c_i \in \mathcal{M}_c$ . First we show that, as with application of Rule 2 on the least significant pairs, the assumptions and implied equalities are insufficient to show equality in the most significant

pairs of constraints of a larger arity.

The set of assumed equalities in application of Rule 3' on  $c_{ik}$  is a subset of the set of assumed equalities in the application of Rule 2 on  $c_{ii}$  since  $k < i$ . As such, the most significant pairs in constraints  $c_b \in \mathcal{C}'$  for  $b > i$  are never assumed to be equal.

We now show that the constraints  $c_a \in \mathcal{C}'$  for  $a < i$  do not imply equality in  $c_{ik}$ .

Consider the case where  $i \leq (n/2)$ . Here the equality class containing  $x_1$  is  $\{x_1, x_i\}$  and as such we can only consider the most significant pairs in each constraint, and  $x_k \leq x_{k+i}$  cannot be removed.

Now consider where  $i > (n/2)$ . Since all variables on any one side of a lex constraint are distinct, and since in  $\mathcal{M}_c$  any variable occurs on the left hand side of the constraint before it occurs on the right hand side, the variable  $x_k$  is never assumed to be equal to any other variable in the application of Rule 3'. Therefore, to show  $x_k = x_{k-(n-i)}$  we require  $x_k$  to appear on the left hand side of another constraint.

The variable  $x_k$  appears on the left hand side of each constraint  $c_a$  for  $a < i$ , but only after the variable  $x_i$  has appeared on the right hand side. The variable  $x_i$  only appears in the least significant pair of  $c_i \in \mathcal{M}_c$  therefore  $x_i$  is never assumed equal and we cannot reach the pair in  $c_a$  containing  $x_k$ .

There exists no activation chain with goal  $x_k = x_{k-i}$  or  $x_k = x_{k-(n-i)}$  for Rule 3', so none of these pairs can be removed by Rule 3'.

It follows that  $\mathcal{M}_s$  is the unique fixpoint and hence the system is confluent.  $\square$

In both cases the general formulae produce a linear number of constraints. We intend to prove confluence results for all groups and constructions in [Grayland *et al.*, 2008].

## 6 An Algorithm to Detect Blocks

In this section we discuss an algorithm  $\mathcal{D}$  to detect blocks and hence decide if a set of lex constraints will reduce confluent to a set of size  $m$ . The algorithm utilises the Rule 2 and 3' reduction algorithm  $\mathcal{R}$  discussed in [Öhrman, 2005] and [Grayland *et al.*, 2008]. The time complexity of  $\mathcal{D}$  is only a factor of  $m$  greater than the time complexity of  $\mathcal{R}$  since it utilises at most  $m$  copies of that procedure.

Given a set of lex constraints  $\mathcal{C}$  and the Rule 2 and 3' reduction  $\mathcal{R}$ , the following algorithm detects blocks and decides if a reduction will be confluent. A negative result does not necessarily mean that the reduction is not confluent, we discuss this point towards the end of this section.

1. apply  $\mathcal{R}$  to  $\mathcal{C}$  until some fix-point  $\mathcal{C}'$
2. for each constraint  $c' \in \mathcal{C}'$ 
  - (a) locate the corresponding constraint  $c \in \mathcal{C}$  to  $c'$ .
  - (b) run  $\mathcal{R}$  on  $c'$ , using  $(\mathcal{C} \setminus c)$  as the additional constraints, until some fix-point with constraint  $c''$

TransitiveGroup(6, x)	Result	Time (ms)
1	Confluent	79
2	Confluent	46
3	Confluent	125
4	Confluent	141
5	Undecided	94
6	Undecided	172
7	Undecided	172
8	Confluent	453
9	Undecided	297
10	Undecided	406
11	Undecided	219
12	Confluent	718
13	Undecided	328
14	Confluent	1375
15	Confluent	2657

Figure 3: Results from running  $\mathcal{D}$  on 15 lex leaders derived from the first 15 transitive groups on 6 points in GAP’s library.

(c) if  $c' \neq c''$  then END return Undecided

3. END return Confluent

**Theorem 5** *When applied to a set of lex constraints  $\mathcal{C}$ , if the algorithm  $\mathcal{D}$  returns Confluent then the reduction is confluent.*

**Proof**

It is clear that  $\mathcal{R}$  produces a minimal set, since we apply it until a fixpoint is reached. Suppose that  $c'_i \in \mathcal{C}'$  cannot be reduced further by  $\mathcal{C} \setminus c_i$ . Then no matter what order the Rules were applied to reduce  $\mathcal{C}$ , the constraint beginning  $c'_i$  must occur in the fixpoint. If this holds for all  $i$  then there is a unique fixpoint, namely  $\mathcal{C}'$ .  $\square$

Further work is required to complete this algorithm in order that it detects all confluent reductions. For example, if the constraint  $c_i$  could be removed by some other constraint  $c_j$ , where  $c_j$  and  $c_i$  both reduce to the same prefix under the other constraints, then we could say the reduction is still confluent since the resulting set of constraints will always be the same. This is the kind of confluent reduction we find with the symmetric groups.

Figures 3 and 4 show the results of testing this algorithm on various symmetry groups. We can see that there are a number of examples where reduction is confluent. Note also that the case we know to be not confluent, the `TransitiveGroup(6,6)` from Theorem 1, is marked as being undecided by the algorithm, which is what we would expect. We ran further tests on the first 8 Symmetric and Cyclic groups with the algorithm returning confluent in each case.

## 7 Discussion

To illustrate briefly the motivation for using a minimum set of lex constraints we present an example, which shows

PrimitiveGroup(x, y)	Result	Time (ms)
(6,1)	Confluent	1078
(6,2)	Confluent	3219
(6,3)	Confluent	7937
(6,4)	Undecided	15156
(7,2)	Confluent	1313
(7,3)	Confluent	1547
(9,1)	Confluent	49422
(9,2)	Undecided	104515
(9,3)	Undecided	211532
(9,4)	Undecided	210328

Figure 4: Results from running  $\mathcal{D}$  on 10 lex leaders derived from primitive groups in GAP’s library.

that different fixpoints under non-confluent applications of Rules 2 and 3’ can require significantly different times to solve. Consider a problem of 36 variables, each with domain  $\{1, \dots, 5\}$ , and divided into subsets of 6. Each subset is constrained to sum to 26 and to have at most one representative from each equivalence class under the symmetries given in the proof of Theorem 1. We break the symmetry on each subset using the reduced sets of lex constraints given in methods 1 and 2 in the proof of the same theorem. Finding all solutions reduced by method 1 took 8,625s, compared to 7,853s for method 2. This provides evidence that it is worthwhile to find a truly minimum fixpoint in reducing sets of lex constraints. In this example there is only one block, for examples with more blocks we expect a still greater variation in the number and arity of fixpoint constraints, and hence solving time.

The contributions of this work are threefold. Previously to prove confluence, it was necessary to show that all possible orders of applications of the rules produced the same fixpoint, which was infeasible with large examples. Additionally, if a system of lex constraints is confluent with respect to Rules 2 and 3’, then every reduction strategy is optimal. Conversely, if the system is not confluent, then the blocks determine all fixpoints. One next step is to complete the algorithm to detect blocks such that it also detects set-wise confluence, and hence to determine which families of groups produce blocks. We will also investigate optimal reduction strategies for blocks.

## References

[Conway *et al.*, 1998] J. H. Conway, A. Hulpke, and J. McKay. On transitive permutation groups. *LMS J. Comput. Math.*, 1:1–8, 1998.

[Crawford *et al.*, 1996] J. Crawford, M. Ginsberg, E. Luks and A. Roy. Symmetry-Breaking Predicates for Search Problems. *Proc. KR*, 148–159, 1996.

[Frisch *et al.*, 2003] A. M. Frisch and W. Harvey. Constraints for Breaking All Row and Column Symmetries in a Three-by-Two-Matrix. *Proc. Symcon*, 2003.

- [The GAP Group, 2007] The GAP Group, *GAP – Groups, Algorithms, and Programming, Version 4.4.10*; 2007, (<http://www.gap-system.org>).
- [Grayland *et al.*, 2008] A. Grayland, C. Jefferson, I. Miguel, C. M. Roney-Dougal. Minimal ordering constraints for some families of variable symmetries. Circa Preprint 2008/1, 2008.
- [Luks *et al.*, 2004] E. M. Luks and A. Roy The complexity of symmetry-breaking formulas. *Ann. Math. Artif. Intell.*, 41(1):19-45, 2004.
- [Öhrman, 2005] H. Öhrman. *Breaking Symmetries In Matrix Models*. Masters Thesis, Dept. Information Technology, Uppsala University, 2005.
- [Puget, 2005] J-F. Puget. Breaking Symmetries In All Different Problems. *Proc. IJCAI*, 272–277, 2005.
- [Roy, 2007] A. Roy Symmetry-breaking formulas for groups with bounded orbit projections. *SymCon '07*, 2007.