Confluence of Reduction Rules for Lexicographic Ordering Constraints

Andrew Grayland[1], Ian Miguel[1] and Colva M. Roney-Dougal[2]
(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 CSPs typically produces a large set of lexicographic ordering constraints. Several rules have been proposed to reduce such sets whilst preserving logical equivalence. These reduction rules are not generally confluent: they may reach more than one fixpoint, depending on the order of application. These fixpoints vary in size. Smaller sets of lex constraints are desirable so ensuring reduction to a global minimum is essential. We characterise the systems of constraints for which the reduction rules are confluent in terms of a simple feature of the input, and define an algorithm to determine whether a set of lex constraints reduce confluently.

Introduction

Constraint models often contain *symmetries*, partitioning the set of assignments into equivalence classes. Symmetries can be exploited by restricting search 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 many lex constraints may be required. Symmetries are captured using the mathematical formalism of a group. A group on n points can have up to $n!$ elements, requiring $n!$ lex constraints. A special case was identified in (Puget 2005), where each variable must be assigned a distinct value, in which the set of ordering constraints collapses to just $n-1$ binary inequalities. The papers (Frisch and Harvey 2003), (Öhrman 2005) and (Grayland et al. 2009) 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 `alldifferent` constraint. The advantage of this reduction method over other similar static methods is that the reductions may be applied to any group, producing a new set of lex constraints. In (Luks and Roy 2004) the lex constraints are described as Boolean formulae, and redundant clauses are pruned; this is extended 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 resulting constraints are not lex constraints and an efficient propagation algorithm has not yet been discussed, this paper continues to advance lex constraint reduction method.

In (Grayland et al. 2009) it is shown that a smaller number of equivalent lex constraints produces faster solve times and uses less memory in search when compared to a larger set. At this time it was unknown whether various complete reductions of the same set of lex constraints were possible so the paper requires complex proofs to show that each set of constraints discussed are the minimum reduction, and not some local minimum. This process is time consuming but was necessary in order to best utilise the time savings offered by a reduced set of ordering constraints.

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 without the need for complex proofs. It also goes some way towards directing reductions in the future by highlighting areas where divergence will occur in nonconfluent systems.

Background

Rules 1 and 2 were introduced in (Frisch and Harvey 2003) to reduce the number and arity of lex constraints whilst maintaining logical equivalence. Rule 3, which supercedes and is stronger than Rules 1 and 2, is defined in (Öhrman 2005). Let α, β, γ, and δ be strings of variables, and x and y be individual variables.

1 If $\alpha = \gamma$ entails $x = y$ then a constraint c of the form

$\alpha x\beta \leq_{\text{lex}} \gamma y\delta$ may be replaced with $\alpha\beta \leq_{\text{lex}} \gamma\delta$.

2 If $C = C' \cup \{\alpha x \leq_{\text{lex}} \gamma y\}$ is a set of constraints, and $C' \cup \{\alpha = \gamma\}$ entails $x \leq y$, then C may be replaced with $C' \cup \{\alpha \leq_{\text{lex}} \gamma\}$.

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 $x = y$ (or $C' \cup \{\alpha = \gamma\}$ entails $x \leq y$ where $|\beta| = 0$), then C may be replaced with $C' \cup \{\alpha\beta \leq_{\text{lex}} \gamma\delta\}$.

Definition 1 *Let α, β, γ, and δ be strings of variables, and x and y be individual variables. In a lex constraint of the form $\{\alpha x \leq_{\text{lex}} \gamma y\}$, the pair $x \leq y$ is said to be* the least significant.

Definition 2 *Examining the pair of variables $x \leq y$ of a lex constraint c from a set of constraints C, we say that the* support *required to remove $x \leq y$ is the set of pairs $C \setminus c$ and the more significant pairs from C that entail $x \leq y$.*

Rule 3 extends the previous Rules by allowing 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. It is therefore more convenient to work with Rule 2 and a new 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 $C' \cup \{\alpha\beta \leq_{\text{lex}} \gamma\delta\}$.

As an example of Rule 1, consider a constraint c: $x_1x_2 \leq_{\text{lex}} x_2x_1$. 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 1 we need only consider the first pair of variables, reducing c to $x_1 \leq x_2$ without modifying the set of solutions.

For Rule 2 we can also use the rest of the constraints for the support to remove our pair under consideration. Consider the two constraints $x_1x_2\mathbf{x_4} \leq_{\text{lex}} x_2x_3\mathbf{x_5}$ and $x_1x_4 \leq x_3x_5$ where we are attempting to remove the bold pair $x_4 \leq x_5$. We first assume $x_1 = x_2$ and $x_2 = x_3$ since the least significant pair in this constraint will only be used if all more significant pairs are equal. Using these assumptions we can see that the first pair of the other constraint, $x_1 \leq x_3$ is equal. This means that $x_4 \leq x_5$ is active in that constraint. We can therefore say that $x_4 \leq x_5$ in the first constraint is entailed and it can be removed by Rule 2.

Rule 3′ generally requires a little more reasoning so an example of this in action is included in more detail later in the proof of Theorem 1.

It was demonstrated in (Grayland et al. 2009) that 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 3 *A* reachable fixpoint *for a set $\mathcal{C}$ of lex constraints is a set of lex constraints produced by removing pairs from $\mathcal{C}$ 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. 2009). This means that there may be a number of possible reductions each resulting in a differing solve time.

Definition 4 *A terminating rewriting system is* confluent *if the rewrite rule reaches the same fixpoint irrespective of the order in which they are applied.*

More details on the extensive research already completed in other fields into confluence can be found in (Baader and Nipkow 1998).

Definition 5 *A group of permutations of X is* transitive *if every point in X can be mapped to any other.*

The proof of nonconfluence in (Grayland et al. 2009) relied on a set of lex constraints derived from an *intransitive* group of symmetries. For this paper we are more interested in transitive groups. Although intransitive group symmetries appear in CSPs, it is much more common to find transitive group symmetries. For example, the symmetries of a set or a multiset are transitive. Furthermore every intransitive group is composed of smaller transitive groups.

Rule 1 is confluent, see (Grayland et al. 2009). For the remainder of the paper we utilise Rule 1 as a preprocessing step to further reduction. On the remaining constraints we will use Rule 2 and 3′ for reduction.

Every permutation can be written as a product of disjoint cycles. Consider the list notation permutation $f := [C, A, F, D, G, B, H, E]$. In cycle notation each point goes to the next one in the cycle, and the last point in the cycle is mapped to the first. Fixed points are omitted, so $f = (ACFB)(EGH)$.

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. 2009). We now show that the same holds even for transitive groups.

Theorem 1 *There exist lex constraints for a transitive group that are not confluent for Rules 2 and 3′.*

Proof Let G be `TransitiveGroup(6,6)` in GAP's library (Conway, Hulpke, and McKay 1998; GAP 2007), permuting A, B, C, D, E, F. The group elements are:

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

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

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

Pairs $B \leq A$ in (5) and (6) can both be removed.

Method 1: From (5) by Rule 3′, see 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,see Figure 1. 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, $(1), (2), (3), (4), (5)$. □

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 pairs which can be removed by Rule* 3′ *but not Rule* 2.

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, nonconfluence 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 nonconfluent 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 $\mathcal{C} = \{c_1, \ldots, c_k\}$ 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 6 *Let $\alpha \in 1..k$ and $\beta \in 1..l$ where $l = |c_\alpha|$. A* goal $c_{\alpha\beta}$ *is a pair under consideration for removal by Rules* 2 *and* 3′. *Each goal $c_{\alpha\beta}$ defines an activation graph $G_{\alpha\beta}$, which is a digraph generated from the assumptions made to remove $c_{\alpha\beta}$ 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_α, 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 *(solid edge) if the pair of variables are equal, and is* amber *(dashed edge) otherwise. Inactive nodes are* red *(dotted edge). 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 $A_{ij} \neq \emptyset$ then it entails the equality of $c_{i(j-1)}$. The 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 $c_{i(j-1)} \sim c_{ij}$, node $c_{i(j-1)}$ 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 7 *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 an argument for the removal of $c_{\alpha\beta}$. To clarify this, we present activation graphs G_{52} and G_{62} for the set $\{(1), \ldots, (6)\}$ 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 now modify C_3. Remove c_{ij}, then add all edges from G_{st} required to activate all of A_{ij}, then place a

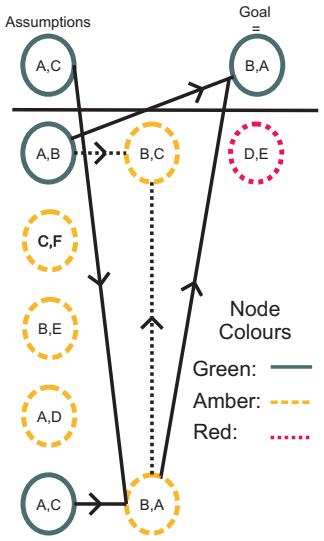

Figure 1: This gives the reduction described in Method 1 of Theorem 1. The solid black edges represent a possible activation chain, with the dashed edges representing activations not used in that activation chain. Green nodes are equal, amber nodes are less than or equal and red nodes are not active.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

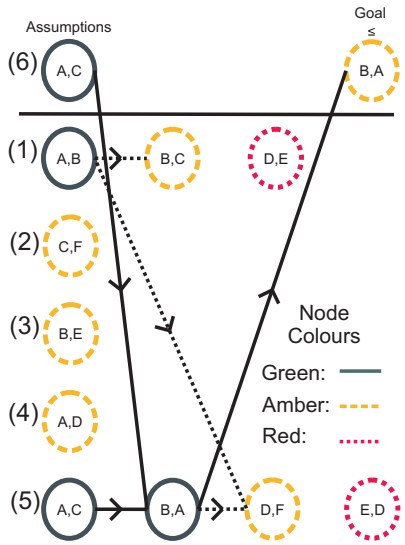

Figure 2: This gives the reduction described in Method 2 of Theorem 1. The solid black edges represent a possible activation chain, with the dashed edges representing activations not used in that activation chain. Green nodes are equal, amber nodes are less than or equal and red nodes are not active.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

directed edge from every node in A_{ij} to every node with 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 $c_{ij} \notin C_3$. □

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

Definition 8 *A* block B *is a set of at least two pairs* c_{ij}*, where not all of the constraints* c_i *have the same fixpoint if reduced first, such that for all* $p, q \in B$ *the node* q *is in at least one activation chain for* p*, and conversely all chains for* p *contain at least one* $q \in B$*.*

Our main result shows that a system of lex constraints is confluently reduced if and only if it contains no blocks.

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

Proof First we show that blocks prevent confluence. Since every element of a block B has at least one activation chain, any element of B can be removed. However, the last element of B will have no remaining activation chains – all such chains contain at least one member of B. 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 $3'$.

We sketch the 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 C with $c_{ij} \notin C$ 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. □

Some Families of Symmetries

In (Grayland et al. 2009) a number of general formulae for the construction of minimal lex constraints for problems with particular symmetries were defined. These symmetries are amongst the most common in CSPs. 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.

Symmetric Groups

The symmetric group, S_n, is the group whose elements are the set of bijections from $\{1, \ldots, n\}$ into itself; we can freely interchange all variables. Symmetric groups arise frequently as symmetries of CSPs, in particular whenever a set is modelled as a list we introduce the

symmetric group on variables. Since the set is unordered but a list is ordered we can freely interchange any variables of the list and get a new list representing the same set.

Theorem 3 *Let P be a CSP with n decision variables $\{x_1, ..., x_n\}$ whose symmetry group is S_n on variables. Given a lex leader with variable ordering x_1 to x_n, Rules 2 and 3′ reduce the corresponding lexicographic constraints confluently to:*

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

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

Each pair in $\mathcal{M}_s$ is the most significant non-trivial pair of one or more lex constraints from the full set, $\mathcal{C}$. Rule 1 confluently removes all trivial pairs $x_i = x_i$ (Grayland et al. 2009). 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 jth 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 $\mathcal{C}_1' = \mathcal{C}_1 \backslash c_i$, 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 $x_i \leq x_j \leq \ldots \leq x_k \leq x_{i+1}$.

Given our canonical ordering, the left hand side of each constraint in $\mathcal{C}$ is $x_1 x_2 \ldots x_n$. 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 contains both $x_i \leq x_b$ and $x_c \leq x_{i+1}$. 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 $x_b \leq ... \leq x_c$ 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}. No assumptions are 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 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.

Thus the Rules are confluent. □

Cyclic Groups

If all elements of a group G can be written as powers of some fixed $g \in G$ then G is *cyclic*. The cyclic group on n points is denoted C_n.

Theorem 4 *(Grayland et al. 2009)*

Let P be a CSP with decision variables, $\{x_1, x_2, \ldots, x_n\}$ whose symmetry group is the cyclic group C_n on variables. A complete set $\mathcal{M}_c$ of symmetry breaking constraints is:

$$\begin{array}{lrll} i = 1 & x_1 & \leq & x_2 \\ i = 2 & x_1 x_2 & \leq_{\text{lex}} & x_3 x_4 \\ \vdots & & \vdots & \\ i = n-1 & x_1 x_2 \ldots x_{n-1} & \leq_{\text{lex}} & x_n x_1 \ldots x_{n-2} \end{array}$$

Lemma 2 *Let $\mathcal{M}_c$ be as above. If $i > n/2$ then the assumption of equality in the first $(i-1)$ pairs in c_i partitions the variables in c_i into $n-i$ nontrivial equality classes. If $i \leq n/2$ then $(i-1)$ nontrivial equality classes are produced.*

Proof The first case is $i \leq n/2$. Then $c_i \in \mathcal{M}_c$ is $x_1 \ldots x_i \leq_{\text{lex}} x_{i+1} \ldots x_{2i}$. All variables in c_i are distinct, hence $i-1$ nontrivial equality classes are created.

We now assume that $i > n/2$, so c_i is

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

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

We assume that $x_1 = x_{i+1}$, $x_2 = x_{i+2}$, ..., $x_{n-i} = x_n$. After these $n-i$ pairs of variables there is a repeat on the right hand side of $x_1 x_2 x_3 \ldots x_{(n-i)}$. We assume that $x_1 x_2 x_3 \ldots x_{n-i} = x_{(n-i)+1} x_{(n-i)+2} \ldots x_{2(n-i)}$.

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

As each equality class grows, the additions are new variables, so the initial $n-i$ classes do not merge. The new variables are always assumed to be equal to a variable currently in a class of size at least 2, so there will never be more than $n-i$ classes of size at least 2.□

Theorem 5 *Let P be a CSP on variables $\{x_1, ..., x_n\}$ with symmetry group C_n on variables. Given a lex leader with the natural variable ordering, Rules 2 and 3′ reduce the corresponding lex constraints confluently to $\mathcal{M}_c$.*

Proof We show that there is only one reachable fixpoint. The complete set of lex constraints $\mathcal{C}$ for C_n is:

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

We show that $\mathcal{C}' = \mathcal{C} \setminus c_i$ does not support the removal of c_{ii} by Rule 2. We then show that $\mathcal{C}'$ does not remove any pair from c_i by Rule 3′. The result follows.
RULE 2: The pair c_{ii} is $x_i \leq_{\text{lex}} x_j$, 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 $x_1 \dots x_{i-1} = x_{i+1} \dots x_n x_1 \dots x_{2i-n}$ if $2i > n$.

We begin by showing that we cannot activate any pair with x_i on the left hand side (LHS) and that x_i is not equal to any other variable, therefore we cannot imply that $x_i \leq x_j$. Our arguments depend on whether $2i \leq n$.

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 $x_1 \leq x_{i+2}$, $x_1 \leq x_{i+3}$, …, $x_1 \leq x_{n-1}$. 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 x_i is not assumed to be less than or equal to anything c_{ii} cannot be reduced.

Now assume that $2i > n$, so that c_{ii} is $x_i \leq x_j$, where $j = 2i - n$. Look for x_i on the LHS of 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 $x_1 = x_{i+1}$, $x_2 = x_{i+2}$, …, $x_{n-i} = x_n$.

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

Now consider c_a for $a < i$. There is equality in the most significant pairs of $c_{n-i}, c_{2(n-i)}, \dots$, because the LHS of the pair is x_1 which is equal to every $(n-i)^{th}$ element by Lemma 2. In these constraints the pairs of variables are matched to those in the Rule-2-assumed equality classes from c_i. This is because the equality relations step along in groups of $n - i$.

The first pair of variables in c_a which are not assumed to be equal is the pair with x_i on its RHS, hence at most we deduce that x_i is greater than another variable. This does not imply equality in the most significant pair of any other constraint, so we still cannot use later pairs. Since x_i has not been assumed to equal anything else and since nothing has implied it to equal anything else, we cannot deduce that $x_i \leq x_j$.
RULE 3′: We first consider Rule 3′ on the pair c_{ik} for $k < i$, which is not least significant in the constraint $c_i \in \mathcal{M}_c$. First we show that 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 $k < i$. Therefore 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}. First consider $i \leq (n/2)$. The equality class containing x_1 is $\{x_1, x_i\}$ so we can only consider c_{a1}, and $x_k \leq x_{k+i}$ cannot be removed.

Now consider $i > (n/2)$. All variables on one side of c_a are distinct, and variables occur on the LHS of c_i before the RHS, so x_k is not assumed to be equal to any other variable under Rule 3'. To show $x_k = x_{k-(n-i)}$ we require x_k to appear on the LHS of another constraint.

The variable x_k appears on the LHS of c_a after x_i has appeared on the RHS. The variable x_i only appears in the least significant pair of c_i, so x_i is never assumed equal to anything and we cannot reach the pair in c_a containing x_k. Thus there exists no activation chain with goal $x_k = x_{k-i}$ or $x_k = x_{k-(n-i)}$ for Rule 3′.

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

In both cases we produce a linear number of constraints. We intend to prove confluence results for all groups and constructions in (Grayland et al. 2009).

An Algorithm to Detect Blocks

We now present an algorithm $\mathcal{D}$ to detect blocks and hence decide if a set of lex constraints will reduce confluently to a set of size m. We use the Rule 2 and 3′ reduction algorithm $\mathcal{R}$ discussed in (Öhrman 2005) and (Grayland et al. 2009). 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 reduction $\mathcal{R}$, the following algorithm detects blocks. A negative result does not necessarily mean that the reduction is not confluent, we discuss this later.

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''
   (c) if $c' != c''$ then END return Undecided
3. END return Confluent

Theorem 6 *Algorithm $\mathcal{D}$ is correct.*

Proof Algorithm $\mathcal{R}$ produces a minimal set, since we apply it until a fixpoint is reached. Suppose that $c'_i \in \mathcal{C}'$ is fixed 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 $\mathcal{C}'$. □

Number	Result	Time	Number	Result	Time
1	C	79	9	U	297
2	C	46	10	U	406
3	C	125	11	U	219
4	C	141	12	C	718
5	U	94	13	U	328
6	U	172	14	C	1375
7	U	172	15	C	2657
8	C	453			

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

Further work is required to detect all confluent reductions. For example, if the c_i could be removed by c_j and c_j and c_i both reduce to the same prefix under the other constraints, then the reduction is still confluent since the resulting set of constraints will always be the same. This is what happens for the symmetric groups, although Theorem 3 shows that in fact the lex constraints for S_n do reduce confluently.

Algorithm $\mathcal{D}$ was tested on various families of groups. The first family selected was the transitive groups on 6 points. The reasons for selecting this family were twofold: the intransitive groups are composed of smaller transitive groups making transitive groups more interesting to examine; and the transitive groups of order 6 are fully classified and relatively small in number allowing for a much fairer distribution of results when compared to picking random groups. In Figure 3 the number i refers to `TransitiveGroup(6, i)` in GAP's library. The second family of groups selected was the primitive groups. The number of transitive groups on more than 6 points grow very rapidly and it is therefore not possible to test all of them in a reasonable amount of time. The primitive groups do not display this behaviour to such a great extent, allowing us to test groups on a range of points. In Figure 4 the pair (x, y) refers to `PrimitiveGroup(x, y)`. The result C means confluent whilst U means undecided. Times are in milliseconds. There are a number of examples where reduction is confluent, thus for each of the groups any fixpoint under Rule 2 and 3′ is the unique minimum. The case we know to be not confluent, the `TransitiveGroup(6,6)` from Theorem 1, is undecided which is what we would expect. We ran further tests on the first 8 Cyclic groups with the answer confluent in each case, again, as expected.

Conclusion

There are three main contributions of this work. Previously it was necessary to show that all possible orders of rule applications produce the same fixpoint to show confluence, which was infeasible with large examples. Now it suffices to find a single fixpoint and use the algorithm to show confluence. Also, 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.

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

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

Acknowledgements:
A. Grayland is supported by a Microsoft Research EPSRC CASE studentship. I. Miguel is supported by a Royal Academy of Engineering/EPSRC Research Fellowship. C. M. Roney-Dougal is partially supported by EPSRC grant number EP/C523229/1.

References

Baader, F., and Nipkow, T. 1998. *Term Rewriting And All That.* New York, NY, USA: Cambridge University Press.

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

Crawford, J.; Ginsberg, M.; Luks, E.; and Roy, A. 1996. Symmetry-breaking predicates for search problems. *Proc. KR* 148–159.

Frisch, A. M., and Harvey, W. 2003. Constraints for breaking all row and column symmetries in a three-by-two-matrix. *Proc. Symcon.*

GAP The GAP Group. 2007. *GAP – Groups, Algorithms, and Programming, Version 4.4.10.* (`http://www.gap-system.org`).

Grayland, A.; Jefferson, C.; Miguel, I.; and Roney-Dougal, C. 2009. Minimal ordering constraints for some families of variable symmetries. *Ann. Math. Artif. Intell.* To appear.

Luks, E. M., and Roy, A. 2004. The complexity of symmetry-breaking formulas. *Ann. Math. Artif. Intell.* 41(1):19–45.

Öhrman, H. 2005. Breaking symmetries in matrix models. MSc Thesis, Dept. Information Technology, Uppsala University.

Puget, J.-F. 2005. Breaking symmetries in all different problems. *IJCAI* 272–277.

Roy, A. 2007. Symmetry-breaking formulas for groups with bounded orbit projections. *SymCon.*