Symmetry Breaking in Constraint Satisfaction

Eugene M. Luks *
Dept. of Computer Science
University of Oregon
Eugene, OR 97403
luks@cs.uoregon.edu

Amitabha Roy *
Dept. of Computer Science
Boston College
Chestnut Hill, MA 02167
aroy@cs.bc.edu

December 2, 2001

Abstract

Symmetry breaking formulas, introduced by Crawford, Ginsberg, Luks and Roy, are extra constraints that are added to an input constraint satisfaction problem before search. They are true of exactly one element (e.g. the lexicographic leader) from each set of symmetrical points in the search space. We study the computational complexity of generating the entire lex leader formula. We show that it is often intractable to generate the entire lex leader formula: indeed, we prove the existence of inputs for which a natural lex leader formula is of exponential size. At the same time, we show that more sophisticated machinery from computational group theory allows us to construct small lex-leader formulas for a large class of interesting groups (solvable groups or more generally groups with bounded composition factors).

1 Introduction

Exploiting symmetries to develop efficient search algorithms has become a standard tool in solving constraint satisfaction problems. In [3], a new technique to exploit symmetries was developed – their paper introduced the concept of "symmetry breaking predicates". Rather than modifying the search algorithm to use symmetries, they used symmetries to modify (and hopefully simplify) the problem being solved. Their approach was to add additional constraints, *symmetry-breaking formulas*, that are satisfied by exactly one member ("the lexicographic leader") of each set of symmetric points in the search space. Thus, instead of having to reformulate each advance in search technology, this method can be used as a preprocessor to existing or future constraint solvers. This technique was extended and applied to practical planning problems by Joslin and Roy[8].

The basic idea underlying this symmetry breaking approach is as follows. Let T be the input constraint satisfaction problem (csp). One constructs an additional set of symmetry-breaking constraints S e.g. constraints that are satisfied by the lexical leader in each set of symmetrical points. Typically S is constructed from the knowledge of a permutation group that acts as symmetries of T. Consider the augmented csp $T \wedge S$. One would expect that with this augmented problem as input, the search algorithm used to find solutions will automatically search non-symmetrical regions of the search space, thus improving efficiency. This observation has been borne out experimentally as described in [3, 8].

*partially supported by NSF grant CCR9820945

One obstacle to this algorithm is that the lex-leader formula S could possibly be of exponential size, making it infeasible to generate the entire formula. Exploration of this computational bottleneck is the central theme of this paper. We show that there are indeed groups of symmetries for which the "natural" lex-leader formula (as defined in Section 3) is of exponential size. Specifically, we show that this happens for very simple classes of abelian groups (elementary abelian 2-groups i.e. groups where each non-identity element has order 2). This exponential size is because of a combinatorial bottleneck which we can formulate in terms of lattices and anti-chains. This has led us to consider a class of combinatorial objects, *Sperner spaces*, a generalization of Sperner families in extremal set theory [5, 13], whose structure is responsible for this exponential lower bound.

However, it is indeed possible, via results by Babai and Luks [1], to write polynomial size formulas for these elementary abelian groups and in fact, for general abelian (even solvable) groups, if we are allowed to make some reasonable assumptions about the input. These formulas are sensitive to the ordering of variables in the input. Furthermore, their formulas are often too large (a high degree polynomial size) to be of any use. Our results, summarized in Theorem 4.1, substantially improve the size of formulas obtained for these groups.

2 Definitions and Notations

Let G be a group. We write $H \leq G$ when H is a *subgroup* of G. If $H \leq G$, then a *right transversal* of H in G is a complete set of right coset representatives of H in G. The group consisting of all permutations of a set Ω is called the symmetric group, denoted by $\mathrm{Sym}(\Omega)$. G is said to act on Ω if there is a homomorphism $\phi : G \to \mathrm{Sym}(\Omega)$. Let $\omega \in \Omega$ and $g \in G$, then ω^g is the image of ω under $\phi(g)$. Also $\omega^G = \{\omega^g | \ g \in G\}$ is the orbit of G that contains ω. The *point stabilizer* of ω is the subgroup $G_\omega = \{g \in G | \, \omega^g = \omega\}$. The *pointwise stabilizer* of $\Delta \subset \Omega$ is $G_{(\Delta)} = \cap_{\delta \in \Delta} G_\delta$. When Ω is ordered as $\omega_1, \omega_2, \ldots, \omega_n$, then $\Omega_i = \{\omega_1, \omega_2, \ldots, \omega_i\}$ and $G_i = G_{(\Delta_i)}$. Let Δ be an orbit of G on Ω.

Groups are input (and output) via generators. We write $G = \langle X \rangle$ when the set $X \subset G$ generates G. Subgroups of $\mathrm{Sym}(\Omega)$ have succinct descriptions in terms of generators : they have generating sets of size $O(|\Omega|)$ [4]. We refer to any standard text (e.g [7]) for basic facts about groups. For permutation groups, we refer to [4].

A *propositional variable* can take on two values, true or false (we write 0 for false, 1 for true) . Let L be a set of propositional variables. *Literals* are variables in L or negations of variables in L. A *clause* is a disjunction of *distinct* literals in L. A *theory* is a conjunction of clauses. A *truth assignment* for a set of variables L is a function $X : L \to \{0, 1\}$. In the usual way, X extends by the semantics of propositional logic to a function on the set of theories over L and by abuse of notation, we will continue to denote the extended function by X. A truth assignment X of L is called a *model* of a theory T if $X(T) = 1$.

The *propositional satisfiability* problem or SAT is the following decision problem: given a theory, decide whether it has a model. This is a canonical example of an NP-complete problem [6].

Let T be a theory. A sub-collection of clauses of T is said to be a *pruning* if it is logically equivalent to the rest. A particular clause is said to be *non-prunable* if it belongs to all prunings (and is said to be *prunable* otherwise).

3 Lex-leader Formulas – Definitions

In this section, we formalize the notion of lex-leader formulas in the context of a permutation group acting on a set of points (Subsection 3.1). We also discuss how lex-leader formulas can be used to augment boolean theories so as to break symmetries in the input as preprocessing step before search (Subsection 3.2).

3.1 Lex-Leader Formulas for Permutation Groups

Let Ω denote the set $\{1, 2, \ldots, n\}$ and $G \leq \mathrm{Sym}(\Omega)$. Let 2^Ω denote the set of functions from Ω to $\{0, 1\}$ (equivalently, 2^Ω is the set of n-bit strings). G acts on 2^Ω via $X \mapsto {}^gX$ for $g \in G$, $X \in 2^\Omega$ where $({}^gX)(i) = X(i^g)$. [1] Under the action of G, 2^Ω breaks up into orbits under the action of G. One can define a natural lexicographic (dictionary) order in 2^Ω: $X < Y$ if $X \neq Y$ and $X(i) < Y(i)$ for the least i such that $X(i) \neq Y(i)$.

Our goal is to write a formula in propositional logic that is true of only one member from each orbit of functions, which we call a *canonical member*. In this paper, we choose the canonical member to be the lexical leader in the orbit, i.e., a function X such that for all $Y \neq X$ in the same orbit, $Y < X$. Formally, a lex-leader formula for G is a boolean formula $\phi_L(G)$ defined over n variables, whose models are lex-leaders in their orbits. Frequently, we will define $\phi_L(G)$ over a larger set of variables and require that the projection of its models in a fixed set of n coordinates are lex-leaders in their G-orbits. It will be clear from the context when we use these extra variables.

For any $X \in 2^\Omega$ define X_i to be the restriction of X to Ω_i, i.e., X_i is an i-tuple consisting of the first i coordinates of X. We will write $\mathrm{Fix}(g, X, i)$ (which effectively means: g fixes the first i bits of X) to mean the boolean formula $({}^gX)_i = X_i$, i.e., the formula $[(X(1) = X(1^g)] \wedge [X(2) = X(2^g)] \wedge \cdots \wedge [X(i) = X(i^g)]$ (we substitute $X(i^g)$ for $({}^gX)(i)$). We use $X(1), X(2), \ldots, X(n)$ as variable names in formulas without any confusion with the function X evaluated at points $1, 2, \ldots, n$. We write $\mathrm{Geq}(g, X, i)$ (equivalently, $X(i)$ is "greater than or equal" to $X(i^g)$) to mean the formula $X(i) \geq X(i^g)$. Notice that $X(i) \geq X(i^g)$ is just a mnemonic for the boolean expression $X(i^g) \rightarrow X(i)$.

We now show how to write a very naive lex-leader formula. Let $g \in G$ and $X \in 2^\Omega$. Consider the following formula

$$\bigwedge_{1 \leq i \leq n} \mathrm{Fix}(g, X, i-1) \rightarrow \mathrm{Geq}(g, X, i) \tag{1}$$

By our definition of lexical order, any X which satisfies Equation (1) has the property that $X \geq {}^gX$. Thus the conjunction of all the formulas associated with all $g \in G$, namely,

$$\bigwedge_{g \in G} \bigwedge_{i=1}^{n} \mathrm{Fix}(g, X, i-1) \rightarrow \mathrm{Geq}(g, X, i) \tag{2}$$

will be true of only the lexical leader of each orbit of functions.

We rewrite Equation 2 and define the lex-leader formula $LL(G)$ as follows:

$$LL(G) = \bigwedge_{g \in G} \bigwedge_{i=1}^{n} C(g, i) \tag{3}$$

where $C(g, i) = \mathrm{Fix}(g, X, i-1) \rightarrow \mathrm{Geq}(g, X, i)$.

[1] It is natural to write this as a "left action", e.g., we have ${}^{g_1 g_2}X = {}^{g_1}({}^{g_2}A)$, whereas expressing the image of X under g_1 by X^{g_1} would lead to the awkward relation $X^{g_1 g_2} = (X^{g_2})^{g_1}$.

Equation (3) could have duplicate clauses. For example, consider the case when $G = \mathrm{Sym}(\{1,2,3\})$. Then $C((1\ 2),1) = C((1\ 2\ 3),1) = (X(1) \geq X(2))$ which means that the clause $X(1) \geq X(2)$ appears twice in Equation 3. Notice that the group elements $(1\ 2)$ and $(1\ 2\ 3)$ both belong to the same right coset of G_1. This intuition allows us to eliminate duplicate clauses: for each i, we include clauses $C(g,i)$ for each coset representative g of G/G_i. This approach can still leave us with $\sum_{i=0}^{n-1} |G/G_{i+1}|$ clauses (which could be of exponential size).

The question is: can we prune LL(G) further? For example, the clause $C((1,3),1) = (X(1) \geq X(3))$ prunes the clause

$$C((1,2,3),2) = \{(X(1) = X(2)) \rightarrow X(2) \geq X(3)\}.$$

While LL(G) might be of exponential size in the input (recall that permutation groups are input via a small set of generators), one might hope to prune it to polynomial size by removing such redundant clauses. But we shall see that this is not the case (Theorem 4.1 (i)).

Example (cyclic groups): Let G be the cyclic group generated by the group element $(1,2,3)$. The group elements in G are $\{(), (1,2,3), (1,3,2)\}$. We show how to construct $LL(G)$ for this group, following Equation 3. When $g = ()$ (i.e. the identity element), $C(g,1) := (X(1) \geq X(1)), C(g,2) := (X(1) = X(1)) \rightarrow (X(2) \geq X(2))$ and $C(g,3) := (X(1) = X(1) \wedge X(2) = X(2)) \rightarrow (X(3) \geq X(3))$. All of these clauses are tautologies and we can omit them from $LL(G)$. For $g = (1,2,3)$ we have $C(g,1) := X(1) \geq X(2), C(g,2) := (X(1) = X(2)) \rightarrow (X(2) \geq X(3))$ and $C(g,3) := (X(1) = X(2) \wedge X(2) = X(3)) \rightarrow (X(3) \geq X(1))$. Observe that $C(g,3)$ is a tautology and can be removed. We can similarly write $C(g,1), C(g,2)$ and $C(g,3)$ when $g = (1,3,2)$. Thus

$$\begin{aligned} LL(G) \quad &= (X(1) \geq X(2)) \wedge \{(X(1) = X(2)) \rightarrow (X(2) \geq X(3))\} \\ &\wedge (X(1) \geq X(3)) \wedge \{(X(1) = X(3)) \rightarrow (X(2) \geq X(1))\}. \end{aligned}$$

where we have removed the trivial clauses. To see that this is indeed a lex-leader formula, observe that the satisfying assignments are $\{(0,0,0), (1,0,0), (1,1,0), (1,1,1)\}$ which are the lexical leaders in their orbits.

Example of Pruning: Let $G = \mathrm{Sym}(\Omega)$ where $\Omega = \{1,2,\ldots,n\}$. Since the formula in Equation 3 involves a conjunction over every group element, $LL(G)$ has $O(n!)$ clauses. We show $LL(G)$ can be pruned to $O(n)$ clauses.

Consider any permutation $g \in G$. The clauses associated with this element (i.e. which imply that $X \geq {}^g(X)$) are $C(g,1) \wedge C(g,2) \wedge C(g,3) \wedge \cdots \wedge C(g,n)$. Note that any *non-trivial* clause

$$C(g,i) = ((X(1) = X(1^g)) \wedge (X(2) = X(2^g)) \wedge \cdots \wedge (X(i-1) = X((i-1)^g))) \rightarrow (X(i) \geq X(i^g))$$

is pruned by a clause $C(h,i)$ where $h = (i, i^g)$. Thus after pruning all these clauses from $LL(G)$, the only clauses that remain are of the form $X(i) \geq X(j)$ for $i < j$. This means that there are $O(n^2)$ clauses in $LL(G)$ after pruning. But we can further prune by replacing any 3 clauses of the form $(X(i) \geq X(j)) \wedge (X(j) \geq X(k)) \wedge (X(i) \geq X(k))$ by $(X(i) \geq X(j)) \wedge (X(j) \geq X(k))$. This reduces the number of non-prunable clauses to $O(n)$ where each clause of the form $X(i) \geq X(i+1)$, for $i = 1,2,\ldots,n-1$. It is obvious that one cannot prune any other clauses from this collection.

3.2 Symmetry Breaking Formulas

Let T be a theory over an n variable set L. A permutation $g \in \mathrm{Sym}(L)$ is said to be an automorphism (also called a "symmetry") of the theory T if g maps models of T to models and non-models to non-models. The set of all symmetries of a theory is easily seem to form a group: this group is called the "symmetry group"

of the theory, denoted by $\mathrm{AUT}(T)$. Our input will be T and a *subgroup* G of $\mathrm{AUT}(T)$ which is available somehow. Our goal will be to use the presence of this group to find models of T efficiently.

We remark that this is a slight departure from the methodology of [3] which explicitly computed the group of "syntactic symmetries" of an input theory T and only considered such symmetries. These were permutations that permuted the clauses and literals of a theory (by permuting variables) to produce the same theory. Syntactic symmetries can reveal hidden structure in the input problem: e.g. in [8], where the authors considered transportation planning problems, structural symmetries involved intricate switching of packages and destinations which were not obvious.

In this paper, we make no assumptions on how we obtain the input group G. The group G could possibly include symmetries that are not syntactic. For example, G could contain permutations that the user knows are symmetries because of some special knowledge of the domain.

The group $G \leq \mathrm{AUT}(T)$ induces an equivalence relation on the set of truth assignments of L, wherein X is equivalent to Y if $Y = {}^{g}X$ for some $g \in G$; thus, the equivalence classes are precisely the *orbits* of G on the set of assignments. Note, further, that any orbit either contains only models of T or contains no models of T. This indicates why symmetries can be used to reduce search: we can determine whether T has a model by visiting each equivalence class rather than visiting each truth assignment.

More generally, a symmetry breaking formula is chosen so that it is true of exactly one element in each orbit of assignments to variables L in a theory T. We illustrate this with an example:

Example: Let T be $a \vee \overline{c}$, $b \vee \overline{c}$, $a \vee b \vee c$, $\overline{a} \vee \overline{b}$ and let $G = \langle (a\ b) \rangle$. It is clear that $(a\ b) \in \mathrm{AUT}(T)$, in fact it is a structural symmetry. The two models of T are $(1,0,0)$ and $(0,1,0)$ (where the first, second and third coordinates are true/false values of a, b and c respectively). Clearly, this permutation maps models to models. We can "break" this symmetry by adding the clause $b \rightarrow a$ which eliminates one of the models, $(0,1,0)$, leaving us with only one model from the orbit.

In general, we introduce an ordering on the set of variables, and use it to construct a lexicographic order on the set of assignments. We will then add a formula that is true of only the lexically largest model under this ordering, within each orbit.[2] Equation 3 is an example of such a formula.

Remark: The problem of finding syntactic symmetries of T is an interesting problem in its own right and one which we don't address in this paper. This problem is equivalent to the *graph isomorphism problem* (ISO) [2]. The complexity of ISO is one of the outstanding problems in computer science: there are no polynomial time algorithms known to solve ISO and it is also not known to be NP-complete (though there is evidence that it is not NP-complete [9]). The problem of finding automorphisms of theories is dealt with in [3] which also studies the effect of augmenting theories with lex-leader formulas on the performance of search algorithms. Because we do allow arbitrary symmetries, we allow specification of symmetry groups even if we may not be able to do graph isomorphism efficiently.

The problem that we address in this paper is the complexity of generating "small" lex-leader formulas for an input sub-group G of symmetries. Since arbitrary groups can be input as groups of symmetries, we ignore the theories that these groups act on and deal directly with the groups. We prove exponential lower bounds for $\mathrm{LL}(G)$ even when G is restricted to groups with orbits of size 2 (which forces G to be elementary abelian 2-group or in other words, a vector space over $\mathrm{GF}(2)$). However we also show that if we are allowed to add a polynomial number of extra variables, we can write a polynomial size lex-leader formula $\phi_L(G)$ for a large class of groups which also include these elementary abelian 2-groups. We summarize the main results in the next section.

[2] We note that this is surely not the *only* way to create symmetry-breaking formulas. One can break symmetries by adding any formula that is true of one member of each equivalence class.

4 Statement of Results

Our goal is to study the size of $\phi_L(G)$ and LL(G) (Equation 3) for various classes of groups. We emphasize that this goal is not just to prove an exponential lower bound to LL(G) – after all this formula could have numerous identical clauses and also have numerous redundant prunable clauses. Instead, we want to prove a lower bound on the size of the minimal equivalent sub-formula in LL(G), i.e., the number of non-prunable clauses.

Note that any definition of lexical order on the set of assignments presupposes an ordering of the underlying set that G acts on. It is possible that some orderings may lead to a lex-leader formula (as prescribed by 3) with no small equivalent formulae, whereas a different ordering might lead to a more tractable lex-leader formula, though we do not have real examples to exhibit this (see end of Section 5 for a discussion of effect of order). Then any theorem on lex-leader formula assumes an implicit understanding of the order of the underlying set.

We now summarize our results in the next theorem, whose proof is delegated to subsequent sections.

Theorem 4.1 *(i) There exist groups $G \leq Sym(\Omega)$ for which the number of non-prunable clauses in $LL(G)$ is c^n for all possible orderings of Ω, where c is a constant > 1 and $n = |\Omega|$. However, for these groups, there is a lex-leader formula $\phi_L(G)$, of size $O(n^3)$ for any ordering of Ω, that uses $O(n^3)$ additional variables.*

(ii) Let $G \leq Sym(\Omega)$ be an abelian group. Then one can find an ordering of Ω in polynomial time such that there is a lex-leader formula $\phi_L(G)$ of size $O(n^5 \log\log n \log\log\log n)$ ($|\Omega| = n$) defined over a polynomial (in n) number of variables.

In this paper, we give most of the details of the proof of part (i). The proof of the existence of an $O(n^3)$ predicate is shown in Section 6 and we outline the main technique for Theorem 4.1 (ii) in Section 7. To see the details of the proofs of parts (ii) and for extensions of this theorem to some classes of non-abelian groups, the reader is referred to [12].

We emphasize that Theorem 4.1 considers the low-level complexity of generating the lex-leader formula for various classes of groups. This is because a polynomial size formula is already guaranteed by a techniques developed by Babai and Luks in [1]. Specifically, one can conclude the following general statement directly from results in [1]:

Proposition 4.1 *There exists a polynomial size lex-leader formula for groups whose non-abelian composition factors embed in the symmetric group on d points, under certain assumptions on the permutation domain.*

Abelian groups (and in fact, solvable groups) fall in the category covered by Proposition 4.1. This result follows from a polynomial time algorithm developed in that paper for the following problem:

Lex-Leader

Input: $G = \langle S \rangle \leq \mathrm{Sym}(\Omega)$ and $X \in 2^\Omega$ (input as an n-bit string).

Question: Is X the lex-leader in its G-orbit ?

Then Cook's theorem guarantees the existence of a "small" boolean formula which is true of the lex-leaders: this is a polynomial length lex-leader formula.

Our contribution in this paper, summarized in Theorem 4.1, improves their result by providing formulas of (much) smaller size.

5 Exponential Lower Bounds for Lex-Leader Formulas

In this section, we exhibit an exponential lower bound on the size of the "naive" lex-leader formula, LL(G), proving Theorem 4.1 (i).

Given $\Omega = \{1, 2, \ldots, n\}$, $G \leq \mathrm{Sym}(\Omega)$, recall from Equation 3 that the formulas associated with $g \in G$ are $C(g, i)$:

$$[({}^gX)_{i-1} \;=\; X_{i-1}] \rightarrow [X(i) \geq ({}^gX)(i)]\,, \text{ for } i = 1, \ldots, n \tag{4}$$

We now consider the case when n is even and G stabilizes each of the sets $\{2i-1, 2i\}$ for $1 \leq i \leq n/2$. Then G is an elementary abelian 2-group and can be identified with a subspace of Z_2^n as follows

$$g \in G \Leftrightarrow v_g \in V \leq Z_2^{n/2} \text{ where } v_g(i) = 1 \text{ iff } \;(2i-1)^g = 2i$$

where $v_g(i)$ is the ith coordinate of v_g where $1 \leq i \leq n/2$. For $g \in G$, observe that

$$X(2j-1) = ({}^gX)(2j-1) \text{ iff } X(2j) = ({}^gX)(2j)$$

Thus in particular equation (4) is necessarily true when $i^g = i$ or when i is even. If i is odd and $i^g \neq i$ then 4 is equivalent to

$$\left[\bigwedge_{k \leq (i-1)/2,\ (2k-1)^g = 2k} X(2k-1) = X(2k)\right] \rightarrow [X(i) \geq X(i+1)]$$

We say that $C(g, 2i-1)$ is trivial if it is a tautology, i.e., if $(2i-1)^g = 2i-1$.

We remove the clauses that are trivially true from LL(G) to obtain the logically equivalent sub-formula

$$N(G) = \bigwedge_{g \in G} \bigwedge_{\substack{1 \leq i \leq n/2 \\ (2i-1)^g \neq 2i-1}} C(g, 2i-1) \tag{5}$$

Recall that, as in the definition of LL(G), there will be several identical clauses in $N(G)$. However we are ultimately concerned with the number of distinct non-prunable clauses. It suffices to prove an exponential lower bound on the number of non-prunable distinct clauses in $N(G)$.

For $g \in G$, $1 \leq i \leq n/2$, let $v_{g,i} \leq Z_2^i$ where $v_{g,i}(j) = 1$ iff $(2j-1)^g = 2j$ for $1 \leq j \leq i$. $v_{g,i}$ is thus the projection of v_g onto the first i coordinates. If $C(g, 2i-1)$ is non-trivial then clearly $v_g(i) = 1$. For $v, w \in Z_2^k$, $v \preceq w$ iff $v(i) \leq w(i)$ for all $1 \leq i \leq k$. In other words, the order $\preceq$ is the lattice theoretic order defined by set inclusion. For $1 \leq i \leq n/2$, define

$$V_i = \{v_{g,i} \in V \,|\, (2i-1)^g = 2i\}.$$

V_i is a lattice under the partial order defined by set-theoretic inclusion i.e $v \preceq w$ in the partial order iff $v(l) \leq w(l)$ for all $1 \leq l \leq i$. The following lemma gives a combinatorial interpretation of of prunability:

Lemma 5.1 *A clause $C(g, 2i-1)$ in $N(G)$ is non-prunable iff $v_{g,i}$ is minimal in V_i.*

In particular, Lemma 5.1 implies that we never need to compare V_i and V_j for prunability. In particular, Lemma 5.1 provides a bijection between the non-prunable formulas in $N(G)$ and the minimal elements of the lattice V_i. Define $\min(V_i) = \{v \in V_i \,|\, \forall w \in V_i, w \preceq v \rightarrow v = w\}$, i.e., $\min(V_i)$ is the set of minimal elements of V_i. We can thus conclude

Theorem 5.2 *Let $G \cong V \leq Z_2^n$. The number of non-prunable formulas in $N(G)$ is*

$$\eta(V) = \sum_{i=1}^{n} |\min(V_i)|.$$

Henceforth, we will work with these groups in their vector space representation, i.e., as subspaces of Z_2^n for some n. Our goal will be to exhibit subspaces of Z_2^n with exponentially large $|\min(V_n)|$ – these will represent groups with an exponential number of distinct non-prunable clauses. The proof of the following theorem exhibits such a construction.

Theorem 5.3 *There exist subspaces $V \leq Z_2^{2n+1}$ for which*

$$|\min(V_{2n+1})| \geq 2^{n-1}.$$

Proof: Let $G \leq \mathrm{Sym}(\Omega)$ and $G \equiv V(n) \leq Z_2^{2n+1}$. For $S \subseteq \{1, \ldots, n\}$ let $v_S \in V(n) \leq Z_2^{2n+1}$ be defined as follows:

$$v_S(i) \quad = \quad \begin{cases} 1 & \text{if } i \in S \\ v_S(i-n) + |S| \mod 2 & \text{if } n+1 \leq i \leq 2n \\ |S| \bmod 2 & \text{if } i = 2n+1 \\ 0 & \text{otherwise} \end{cases}$$

Set

$$V(n) = \{v_S |\ S \subseteq \{1, \ldots, n\}\}.$$

There are 2^{n-1} elements in V_{2n+1}. We claim that all of them are minimal in V_{2n+1}. To see this, let $v_S, w_{S'} \in V_{2n+1}$ with $v_S \prec w_{S'}$. Clearly, $S \subset S'$, i.e., there exists a j, $1 \leq j \leq n$, such that $v_S(j) = 0, w_{S'}(j) = 1$. Note that since $v_S, w_{S'} \in V_{2n+1}$, $|S| \equiv 1 (\mathrm{mod}\ 2)$ and $|S'| \equiv 1 (\mathrm{mod}\ 2)$. Observe that, by definition, $v_S(j+n) = v_S(j) + |S| \bmod 2 = 1$ and $w_{S'}(j+n) = w_S(j) + |S'| \bmod 2 = 0$, so $v_S(j+n) > w_{S'}(j+n)$. Hence $v_S \not\preceq w_{S'}$: a contradiction. □

We can prove that for groups under consideration, the ordering of variables does not have any significant effect:

Lemma 5.4 *There exist groups $G \leq Sym(\Omega)$ such that the number of non-prunable clauses of $LL(G)$ is at least $2^{\frac{n-3}{2}}$ (where $|\Omega| = n$) for all possible orderings of Ω.*

Lemma 5.4 thus proves the first part of Theorem 4.1 (i).

It is conceivable (but is an open question) that for some groups $G \leq \mathrm{Sym}(\Omega)$ the number of non-prunable elements in $\mathrm{LL}(G)$ will be sensitive to the ordering of Ω. This is illustrated very nicely by the *Lex-Leader* problem defined in Section 4. While this problem is solvable in polynomial time for Γ_d groups when we assume an ordering of the permutation domain, it is NP-hard (and not known to be in NP) for for elementary abelian 2-groups for some orderings of the permutation domain. This result is Proposition 3.1 (which we quote below) from [1] which paper also proved the result in that Lex-Leader is in P (with certain assumptions on the order of the permutation domain) as a "striking counterpoint" to how the order of the permutation domain affects the computational complexity of the lex-leader problem.

Lemma 5.5 *[1, Proposition 3.1] [10] The problem of finding the lexicographic leader in the G-orbit of $X \in 2^\Omega$ for $G \leq Sym(\Omega)$ is NP-hard even if G is restricted to be an elementary abelian 2-group.*

Lemma 5.5 says that some orders are "bad" and we already know that some orders are "good" and that those good orders can be found efficiently. Thus we have the following corollary:

Corollary 5.6 *Unless NP = P, there is no polynomial time algorithm that computes a lex-leader formula $\phi_L(G)$ for an arbitrary group $G \leq Sym(\Omega)$.*

Proof: If such an algorithm existed, it would provide a reduction from the problem in Lemma 5.5 to SAT, hence forcing an NP-hard problem to lie in P, which forces NP = P. □

Corollary 5.6 is not a deterrent to finding one can generate efficient lex-leader formulas for special classes of groups. In fact, Theorem 4.1 (ii) shows how to construct such formulas for abelian groups.

In summary, intractable examples are thus those vector spaces where an exponential number of vectors in one of the lattices for a coordinate are minimal. So if the vector space was such that *all* vectors were incomparable (in the inclusion order) then clearly it will be one of the intractable examples. This generalization leads to the concept of Sperner spaces described below.

Sperner spaces are subspaces $V = \{v_1, \ldots v_{2^m}\}$ of Z_2^n where for all non-zero vectors $v, w \in V$, $v \preceq w \rightarrow v = w$. Sperner subspaces can be easily seen as a generalization of Sperner families (see [5]) in an algebraic setting. The existence of exponentially large Sperner spaces (where $m = \Omega(2^{cn})$) give us an alternative proof of Theorem 5.3 and thus, indirectly, for the lower bound in the first part of Theorem 4.1 (i). Indeed, we can prove that for $n \geq 5$, there are Sperner subspaces of size $2^{n \log(2/\sqrt{3})}$ (see [12], this result is originally due to Miklós[11] who first proved this bound for an equivalent combinatorial object). In the next section, we show how one can write a small lex-leader formula for groups in this class.

6 Polynomial Size Lex-Leader Formulas for Subspaces of Z_2^n

In this section, we show how one can use linear algebra to write short lex-leader formulas for G when G is a subspace of Z_2^n, thus proving the second half of Theorem 4.1 (i). The essential trick is to express lex-leastness as the non-solvability of a system of linear equations and show that short boolean formulas can express non-solvability (Lemma 6.4). Note that one can write a boolean formula (the "solvability formula") which is satisfiable iff a given system of linear equations is solvable. This can be done directly as we do in Lemma 6.2 or by appealing to Cook's theorem (a polynomial time algorithm to solve equations guarantees an equivalent boolean formula of small size, via reduction to SAT).

We begin with some preliminary results. Let $\mathbf{x} = (x_1, x_2 \ldots, x_n)$ and $\epsilon = (\epsilon_1, \epsilon_2, \ldots, \epsilon_n)$ be 2 n-bit vectors from GF(2)^n. Also, let $b \in$ GF(2). Let $E\,(= E(\mathbf{x}, \epsilon, b))$ denote the equation $\sum_{i=1}^{n} \epsilon_i x_i = b$.

Lemma 6.1 *There exists a boolean formula ϕ of size $\Theta(n)$ defined over $3n$ boolean variables (ϵ_i, x_i, b and $n-1$ additional variables) which is satisfiable iff E is solvable.*

Proof: Observe that E is solvable iff the equations $\mu_1 = \epsilon_1 x_1$, $\mu_i = \mu_{i-1} + \epsilon_i x_i$ for $2 \leq i \leq n-1$ and $b = \mu_{n-1} + \epsilon_n x_n$ are simultaneously solvable where $\mu_i, 1 \leq i \leq n-1$ are new variables. This system is solvable iff the boolean formula

$$(\mu_1 \leftrightarrow (\epsilon_1 \wedge x_1)) \wedge \bigwedge_{2 \leq i \leq n-1} (\mu_i \leftrightarrow (\mu_{i-1} \oplus (\epsilon_i \wedge x_i))) \wedge (b \leftrightarrow (\mu_{n-1} \oplus (\epsilon_n \wedge x_n)))$$

is satisfiable ($\oplus$ refers to the exclusive-or operator). $\Box$

Given a system of equations, we can now apply the construction in Lemma 6.1 to each equation.

Lemma 6.2 *Let $Ax = b$ be a system of equations over GF(2) where A is an $m \times n$ matrix. Then there is a boolean formula $\phi(A, b)$, which is satisfiable iff $Ax = b$ is solvable. Furthermore, $\phi(A, b)$ is of size $\Theta(mn)$ and is defined over the $m(2n + 1)$ variables $A(i, j), b_i, x_i$ and $m(n - 1)$ additional variables,*

Remark: Observe that finding a solution to an $n \times n$-system of equations is in polynomial time (via Gaussian elimination). This algorithm runs in time $O(n^3)$. Cook's theorem [6] now guarantees the existence of a boolean formula which encodes the computation path of this algorithm. Clearly such a formula would be asymptotically larger than the formula obtained by Lemma 6.2 (which is $O(n^2)$).

Let $Ax = b$ be a system of equations over GF(2), where A is an $m \times n$ matrix.

Lemma 6.3 *There exists a system of equations $Cx = d$ over GF(2), where C is an $(n + 1) \times m$ matrix, which is solvable iff $Ax = b$ is not solvable.*

Proof: Consider the vector space spanned by the rows for A together with b, i.e., the row space R of the matrix $[A\ b]$. The system of equations $Ax = b$ is not solvable iff the vector $[0\ 0\ \cdots 1]$ is in the linear span of the vectors in R (which corresponds to a system of equations $Cx = d$). $\Box$

Now we write a polynomial length boolean formula that expresses solvability of a system of equations $Ax = b$. Lemma 6.2 and Lemma 6.3 imply the following:

Lemma 6.4 *Let $Ax = b$ be a system of equations over GF(2) (where A is an $m \times n$ matrix). Then there is a boolean formula, $\overline{\phi}(A, b)$, of size $\Theta(mn)$ defined over variables $A(i, j), x_i, b_i$ and $(n + 1)(m - 1)$ additional variables, which is satisfiable iff $Ax = b$ is* not *solvable.*

Remark: Lemma 6.3 is a crucial ingredient of Lemma 6.4. The ability to write a system of equations which is solvable iff $Ax = b$ is not solvable allowed us to express non-solvability as solvability. Since we want a boolean formula which is satisfiable iff $Ax = b$ is not solvable, observe it *does not* suffice to put a negation sign in front of $\phi(A, b)$. Because $\neg\phi(A, b)$ could still be unsatisfiable while $Ax = b$ is not solvable.

Let $G \leq \mathrm{Sym}(\Omega)$ be as described in Section 5, i.e., $G \equiv W \leq Z_2^{n/2}$ be a group acting on n points $[n] = \{1, 2, \ldots, n\}$ where the orbits of G are the sets $\{2i - 1, 2i\}$ for each $1 \leq i \leq n/2$ (after suitable reordering of Ω if necessary). The following proposition relates lex-leastness with equation non-solvability.

Proposition 6.1 *Let $W \leq Z_2^n$ and let $X \in 2^n$. Then there exists a system of linear equations $E_W(X)$ which is non-solvable iff X is the lex-leader under the action of W.*

The proof of this proposition is beyond the scope of this present paper but may be found in [12]. We give an outline here. To see why lex-leastness corresponds to non-solvability, X is lex-least in its orbit, if for all $w \in W$, ${}^wX > X$. In other words, there is no Y such that $Y = {}^wX$ and $Y > X$. We can write the conditions $Y = {}^wX$ and $Y > X$ as a system of linear equations $E_W(X)$ (this is the heart of the proof to Proposition 6.1). Thus X is lex-least iff $E_W(X)$ is not solvable.

Now Lemma 6.4 implies that there is a boolean formula $\phi = \phi(W, X)$ which is satisfiable iff $E_W(X)$ is not solvable. In other words, X is lex-least iff $\phi(W, X)$ is satisfiable; in other words, $\phi(W, X)$ is the desired lex-leader formula. While $\mathrm{LL}(G)$ for some groups of this class was of exponential size for any ordering of Ω, $\phi_L(G)$ (where $G \equiv W$) is of polynomial size: however, as a penalty, we have to use up to $O(n^3)$ extra variables in addition to $X(1), X(2), \ldots, X(n)$.

7 General abelian groups

The results of Section 6 generalize to general abelian groups. We outline the basic idea, details of the construction can be found in [12]. The essential ingredient is identical in both the vector space and the general abelian case situations: we can express the property of lex-leastness as the non-solvability system of arithmetic equations of an algebraic structure. In the vector space situation, this was a system of equations over GF(2). In the general abelian case, this turns out to be a collection of systems of equations over a Z_m-module. Now a generalization of Lemma 6.3 expresses the non-solvability of a system of equations over a module as the solvability of a system of equations. Then, we express solvability as a boolean formula. The size bound for the abelian case comes from implementing fast arithmetic in modules.

References

[1] Laszlo Babai and Eugene M. Luks. Canonical labeling of graphs. In *Proc. 15th ACM Symp. on Theory of Computing*, pages 171–183, 1983.

[2] James Crawford. A theoretical analysis of reasoning by symmetry in first-order logic (extended abstract). In *Workshop notes, AAAI-92 workshop on tractable reasoning*, pages 17–22, 1992.

[3] James Crawford, Matthew Ginsberg, Eugene M. Luks, and Amitabha Roy. Symmetry breaking predicates for search problems. In *Proceedings of the Fifth International Conference on Knowledge Representation and Reasoning (KR '96)*, pages 148–159, 1996.

[4] John D. Dixon and Brian Mortimer. *Permutation groups.* Springer-Verlag, New York, 1996.

[5] Konrad Engel. *Sperner Theory*, volume 65 of *Encyclopedia of Mathematics and its Applications.* Cambridge University Press, 1997.

[6] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness.* W. H. Freeman and Company, New York, 1979.

[7] M. Hall. *The theory of groups.* Chelsea Publishing Company, New York, second edition, 1976.

[8] David Joslin and Amitabha Roy. Exploiting symmetries in lifted csps. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 197–203. American Association for Artificial Intelligence (AAAI), 1997.

[9] Johannes Köbler, Uwe Schöning, and Jacobo Torán. *The graph isomorphism problem: its structural complexity.* Birkhäuser Boston Inc., Boston, MA, 1993.

[10] Eugene M. Luks. Permutation groups and polynomial-time computation. In W. M. Kantor L. Finkelstein, editor, *Groups and Computation, Workshop on Groups and Computation*, volume 11 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 139–175, 1993.

[11] D. Miklós. Linear binary codes with intersection properties. *Discrete Appl. Math.*, 9(2):187–196, 1984.

[12] Amitabha Roy. *Symmetry Breaking and Fault Tolerance in Boolean Satisfiability.* PhD Dissertation, Department of Computer and Information Science. University of Oregon, 2001.

[13] E. Sperner. Ein satz über untermengen einer endlichen menge. *Math. Z.*, 27:544–8, 1928.

[14] H. Wielandt. *Finite Permutation Groups.* Academic Press, New York, 1964.