Total Ordering on Subgroups and Cosets

Alexander Hulpke*
Department of Mathematics
Colorado State University
1874 Campus Delivery
Fort Collins, CO 80523-1874
hulpke@math.colostate.edu

Steve Linton
Centre for Interdisciplinary Research in
Computational Algebra
University of St Andrews
The North Haugh
St Andrews, Fife KY16 9SS, U.K.
sal@dcs.st-and.ac.uk

ABSTRACT

We show how to compute efficiently a lexicographic ordering for subgroups and cosets of permutation groups and, more generally, of finite groups with a faithful permutation representation.

Categories and Subject Descriptors

F [**2**]: 2; G [**2**]: 1; I [**1**]: 2

General Terms

Algorithms

Keywords

permutation group, matrix group, algorithm, total order

1. INTRODUCTION

Searching for objects in a list becomes easier if the list is sorted. This motivates the desire to define a total ordering that can be computed easily. An ordering can also be useful to designate a particular representative within a class that can be identified repeatedly without the need to store it. In classifications, such as [3], this is a useful tool for the elimination of isomorphic representatives: Define a "canonical" representative under conjugation action of some supergroup to be the smallest element in the orbit. If a group is constructed that is not minimal in its orbit, it cannot be this "canonical" representative and can be discarded immediately without the need to do an explicit isomorphism test.

Many common representations of groups offer a natural definition of a total order on their elements. Permutations are naturally sorted by lexicographic comparison of the images of $[1,\ldots,n]$ under the permutations. (The smallest permutation is thus the identity element ( ).) In the symmetric group S_n this comparison of elements takes at most n point comparisons.

In a similar way a total order on a field F induces a total lexicographic order on the n-dimensional row space F^n which in turn induces a total (again lexicographic) order on the set $F^{n\times n}$ of $n\times n$ matrices with entries in F. A comparison of two matrices takes at most n^2 comparisons of field elements.

As a third case we consider the automorphism group G of a (finite) group A. A total order on the elements of A again induces a total order on the elements of G by lexicographic comparison of the images of a generating list L of A. (It can be convenient to take L to be sorted as well. The most "generic" choice is probably $L = A$.) Comparison of two automorphisms will take at most $|L|$ comparisons of images.

The aim of this paper is to show how such an ordering can be extended to subgroups of the group. In the case of permutation groups, the techniques we describe seem to be known in the folklore, although we could find no reference for them. The extensions to more general settings are the main novelty of this paper.

2. ORDERING ON SUBGROUPS

We now assume that G is a finite group on whose elements a total ordering $\ll$ has been established. We write $\min_{\ll}$ to denote the minimal element of a set under this ordering.

We now define an order on subsets of G in the following way:

DEFINITION 1. *Let $X, Y \subseteq G$. We say that $X \prec Y$ if and only if the $\ll$-sorted list of elements of X is lexicographically smaller than the sorted list of elements of Y.*

For example, for permutations $(1,2) \ll (1,2,3,4)$ and thus $[(1,2),(3,4)] \prec [(1,2,3,4),(1,2,4,3)]$.

It is easily seen that $\prec$ defines a total order on subsets of G. We want to compute this ordering $\prec$ efficiently for subgroups and cosets without having to enumerate all elements. For cosets of the same subgroup, this means simply comparison of their minimal elements:

LEMMA 2. *Let $U \leq G$ be a finite subgroup and $g_1, g_2 \in G$. Then $Ug_1 \prec Ug_2$ if and only if $\min_{\ll}(Ug_1) \ll \min_{\ll}(Ug_2)$.*

PROOF. The sorted list of elements of a coset Ug will start with its minimum $\min(Ug)$. If $\min(Ug_1) = \min(Ug_2)$ both

*Supported in part by EPSRC Grant GL/L21013

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ISSAC'03, August 3–6, 2003, Philadelphia, Pennsylvania, USA.
Copyright 2003 ACM 1-58113-641-2/03/0008 ...$5.00.

cosets contain a common element and thus $Ug_1 = Ug_2$. Otherwise the minima must differ and the comparison between the cosets is already decided on the minimal element. □

A method to compute such minimal coset elements in permutation groups will be given by algorithm 7.

DEFINITION 3. *For a subgroup $U \leq G$ we define the* smallest generating list $\mathcal{S}(U) = (u_1, \ldots, u_m)$ *of U as follows:*

$$u_1 = \min_{\ll}(U), \quad u_i = \min_{\ll}(U - \langle u_1, \ldots, u_{i-1}\rangle)(i > 1)$$

until $\langle u_1, \ldots, u_m\rangle = U$ for a suitable m.

In particular, u_1 is the smallest nontrivial element of U.

We note that a comparison $\mathcal{S}(U) \prec \mathcal{S}(V)$ defines a total ordering on subgroups, but this ordering is not necessarily compatible with the ordering as sets:

EXAMPLE 4. *Let*
$G = \langle (1,2,9)(3,4,5)(6,7,8), (1,4,7)(2,5,8)(3,6,9)\rangle \cong 3 \times 3$ *and* $H = \langle (1,2,9)(3,4,5)(6,7,8)\rangle$. *Then*
$\mathcal{S}(G) = [(1,2,9)(3,4,5)(6,7,8), (1,3,8)(2,4,6)(5,7,9)]$ *and* $\mathcal{S}(H) = [(1,2,9)(3,4,5)(6,7,8)] \prec \mathcal{S}(G)$. *But the sorted element lists start with*

$$\begin{aligned} G &= \{(), (1,2,9)(3,4,5)(6,7,8), (1,3,8)(2,4,6)(5,7,9), \ldots \\ H &= \{(), (1,2,9)(3,4,5)(6,7,8), (1,9,2)(3,5,4)(6,8,7)\} \end{aligned}$$

and therefore $G \prec H$.

By a slight modification it is possible, however, to get an ordering which is compatible:

DEFINITION 5. *Let $U, V \leq G$ and let $u = \max_{\ll}(U)$, $v = \max_{\ll}(V)$, $s = \mathcal{S}(U)$, $t = \mathcal{S}(V)$, $x = |s|$ and $y = |t|$. We define $\widehat{\mathcal{S}}$ by padding the shorter list with the largest element of the respective group:*

$$\widehat{\mathcal{S}}(U) = \begin{cases} \mathcal{S}(U) \sqcup (\underbrace{u, \ldots, u}_{y-x \text{ times}}) & \text{if } x < y \\ \mathcal{S}(U) & \text{if } x \geq y \end{cases}$$

and

$$\widehat{\mathcal{S}}(V) = \begin{cases} \mathcal{S}(V) \sqcup (\underbrace{v, \ldots, v}_{x-y \text{ times}}) & \text{if } x > y \\ \mathcal{S}(V) & \text{if } x \leq y \end{cases}$$

where $\sqcup$ denotes the concatenation of sequences. (The definition thus depends on the pair U, V).

Comparison of the extended generating systems $\widehat{\mathcal{S}}$ now is compatible with the set-wise ordering:

LEMMA 6. *Let $U, V \leq G$, then $U \prec V$ if and only if $\widehat{\mathcal{S}}(U) \prec \widehat{\mathcal{S}}(V)$.*

PROOF. Let $\widehat{\mathcal{S}}(U) = (a_1, \ldots, a_k)$, $\widehat{\mathcal{S}}(V) = (b_1, \ldots, b_k)$, and enumerate the elements of U and V in ascending order: $u_1 \ll u_2 \ll \cdots \ll u_{|U|}$ and $v_1 \ll v_2 \ll \cdots \ll v_{|V|}$.

Now suppose that $U \prec V$, which means (by the definition of lexicographic ordering) there is an i such that $u_j = v_j$ for $j < i$ and either $i > |U|$ or $u_i \ll v_i$. Let l be the largest index such that a_l is among the u_j for $j < i$. Thus, by the definition of $\mathcal{S}$, we have $a_j = b_j$ for $j \leq l$. As $U \neq V$, $\mathcal{S}(U)$ must differ from $\mathcal{S}(V)$, so $l < k$. We note that all the u_j and v_j for $j < i$ must be in $\langle a_1, \ldots, a_l\rangle$ (otherwise l would not have been maximal).

Assume first, that $i > |U|$. Then $U = \langle a_1, \ldots, a_l\rangle$ contains no further elements. Thus a_{l+1} is a padding element and b_{l+1} must be non-padding. Furthermore b_{l+1} must be larger than the largest element of $U = \langle a_1, \ldots, a_l\rangle$, otherwise it would have occurred as a v_j at an index j before $|U|$. Thereby $\widehat{\mathcal{S}}(U) \prec \widehat{\mathcal{S}}(V)$.

Now assume that $u_i \ll v_i$. Then, by the choice of i, u_i can not be contained in the subgroup $\langle a_1, \ldots, a_l\rangle \leq U \cap V$. By the maximality of l, u_i is the smallest element not in $\langle a_1, \ldots, a_l\rangle$. Therefore we have $u_i = a_{l+1}$. If b_{l+1} is not a padding element, it must equal one of the v_j for $j \geq i$, so $a_{l+1} = u_i \ll v_i \leqq v_j = b_{l+1}$ and $\widehat{\mathcal{S}}(U) \prec \widehat{\mathcal{S}}(V)$.

If b_{l+1} is padding, b_{l+1} is the largest element of V and every $v_i \in V$ is smaller or equal. Thus $a_{l+1} = u_i \ll v_i \leqq b_{l+1}$ which proves again $\widehat{\mathcal{S}}(U) \prec \widehat{\mathcal{S}}(V)$.

Conversely, suppose that $\widehat{\mathcal{S}}(U) \prec \widehat{\mathcal{S}}(V)$. Then there is an index l such that $a_j = b_j$ for $j \leq l$ and $a_{l+1} \ll b_{l+1}$. Then $\langle a_1, \ldots, a_l\rangle = \langle b_1, \ldots, b_l\rangle \leq U \cap V$.

Assume first, that a_{l+1} is not padding. Then it is the smallest element of U not in $\langle a_1, \ldots, a_l\rangle$. Let i be the index of this element in U, $a_{l+1} = u_i$. Then the $a_j = b_j$ for $j \leq l$ are among the u_j $(j < i)$ and by the definition of $\mathcal{S}$, we have that $u_j = v_j$ for $j < i$.

If b_{l+1} is not padding it is the smallest element of V not in $\langle a_1, \ldots, a_l\rangle = \langle b_1, \ldots, b_l\rangle$, therefore $v_i = b_{l+1}$. Furthermore, by assumption $u_i = a_{l+1} \ll b_{l+1} = v_i$. Thus $U \prec V$.

If b_{l+1} is padding, $v_i \in V = \langle b_1, \ldots, b_l\rangle = \langle a_1, \ldots, a_l\rangle < U$. As $a_{l+1} \notin \langle a_1, \ldots, a_l\rangle$, we have that $u_i \neq v_i$. Furthermore as $u_j = v_j$ for $j < i$ we must have that $v_i = u_k$ for a $k > i$ and thus $u_i \ll u_k = v_i$. Therefore $U \prec V$.

If on the other hand a_{l+1} is padding, $U = \langle a_1, \ldots, a_l\rangle$ and b_{l+1} (which cannot be padding as well) must be the smallest element of V that is not in U. But it is larger than the largest element a_{l+1} of U, thus the sorted element list of V starts with the full list of elements of U and thus again $U \prec V$.

□

We now show how to compute $\mathcal{S}(U)$ and $\max_{\ll}(U)$ without listing all the elements of U, thus enabling comparison of subgroups according to lemma 6.

3. PERMUTATION GROUPS

We first consider the case of permutation groups. The element order is inherited from an ordering of the permutation domain, two permutations are compared, by comparing the lists of images of the ordered permutation domain lexicographically.

The basic ideas for comparing subgroups already are found in [6, 2]:

For a finite permutation group G acting on the points $1 \ldots, d$, we compute a series of iterated stabilizers

$$\begin{aligned} G \geq G_{[1]} &= \mathrm{Stab}_G(1) \geq G_{[1,2]} = \mathrm{Stab}_{G_{[1]}}(2) \geq \cdots \\ &\geq G_{[1,2,\ldots,n]} = \langle 1\rangle. \end{aligned}$$

We now eliminate from $[1, 2, \ldots, n]$ those points i such that $G_{[1,\ldots,i]} = G_{[1,\ldots,i-1]}$. The resulting set $B_k(G) = (\beta_1, \ldots, \beta_k)$ is still a base of G (see [6] for the definition), it is, in fact, the

lexicographically smallest irredundant base of G. We call the corresponding stabilizer chain $G = G_1 > G_2 > \cdots > G_k > G_{k+1} = \langle 1 \rangle$ the lexicographically smallest reduced stabilizer chain.

Using for $\ll$ the lexicographic comparison of images, we note that every element in $G_{[\beta_1,\ldots,\beta_i]}$ is smaller than any element in $G_{[\beta_1,\ldots,\beta_{i-1}]} - G_{[\beta_1,\ldots,\beta_i]}$. Therefore the smallest generating set $\mathcal{S}(G_{[\beta_1,\ldots,\beta_{i-1}]})$ is obtained from $\mathcal{S}(G_{[\beta_1,\ldots,\beta_i]})$ by appending further elements. (This automatically makes $\mathcal{S}(G)$ a strong generating set.)

Using this stabilizer chain we first show how to determine minimal coset representatives.

ALGORITHM 7. (Smallest Coset Element, [6]) Let $g \in S_n$. The following procedure computes the minimal element in the right coset Gg.

1. [*Initialization*] Let $i := 1$ and $rep := g$.
2. [*Images of* β_i] Let $orb := \beta_i^{G_i}$ and let $img := \{\omega^{rep} \mid \omega \in orb\}$. (These are all potential images of β_i under $G_i rep$.)
3. [*Minimum image*] Let $\mu = \min(img)$ and $\omega = \mu^{rep^{-1}}$. Take $t \in G_i$ such that $\beta_i^t = \omega$. (t can be obtained from the transversal of G_{i+1} in G_i in the stabilizer chain.) Let $rep := t \cdot rep$. (Thus β_i^{rep} is minimal possible.)
4. [*Step*] Let $i := i + 1$. If $G_i \neq \langle 1 \rangle$ then go to step 2. Otherwise return rep.

PROOF. As g is left-multiplied by elements of G, rep lies in the same coset as g. Furthermore in step 3 rep is changed only by elements of G_{i-1}, thus (after i has been increased in step 4) the image β_i^{rep} remains unchanged, and so is minimal possible as it has been chosen in step 2. □

REMARK 8. *If we replace μ in step 3 by the maximum, we obtain the largest coset element. Applying this to the group G itself thus yields the largest element of G needed for padding in lemma 6.*

Since $|orb| < n$, the runtime is dominated by the multiplication of permutations in step 3 and thus essentially the same as for a membership test in G; for base length k and n points, $\mathcal{O}(n \cdot k)$ multiplications are necessary, giving a time complexity $\mathcal{O}(n^2 k)$. (If the transversal is stored in factored form, further multiplications are required.)

REMARK 9. *We can use smallest coset elements also to obtain an element test for double cosets (and to test equality of double cosets): If UgV is a double coset, we compute the smallest elements for all the U-cosets in UgV. We can do this by computing the smallest element g' in Ug, and then computing the orbit of g' under right multiplication by V, always converting a product in the smallest element in its U-coset.*

Then x is in U if the smallest element of the coset Ux is among these U-coset representatives.

Note, however, that the computation of the orbit of g' is not necessarily polynomial time. Indeed equality of double cosets is a known hard problem [4]

Use of algorithm 7 enables us to compute $\mathcal{S}(G)$. (This method surely has been known in folklore for a long time, however the authors have been unable to find an explicit literature reference for it.)

ALGORITHM 10. (Smallest Generating Set for smallest basis) Assume that $\{G_i\}_{i=1}^k$ forms the lexicographically smallest reduced stabilizer chain of G. The following procedure computes $\mathcal{S}(G)$:

1. [*Initialization*] Let $i := k$ and $L := []$.
2. [*Orbit Reached?*] Let $orb := \beta_i^{G_i} - \beta_i^{\langle L \rangle}$. If $orb = \emptyset$ then go to step 4.
3. [*Minimal Element*] Let $\omega := \min(orb)$ and take $t \in G_i$ (from the transversal) such that $\beta_i^t = \omega$. Using algorithm 7 compute the minimal element x of $G_{i+1}t$. (This uses $G_{i+1} > \cdots > G_k$ as a stabilizer chain for G_{i+1}.) Add x to L. Go to step 2.
4. [*Step*] Let $i := i - 1$. If $i > 0$ then go to step 2. Otherwise return $\mathcal{S}(G) := L$.

PROOF. As $\mathcal{S}(G)$ is obtained from the $\mathcal{S}(G_i)$ it is sufficient to show that steps 2 and 3 compute $\mathcal{S}(G_i)$ from $\mathcal{S}(G_{i+1})$. The condition in step 2 ensures that the elements in l generate the full orbit of β_i, thus together with G_{i+1} they generate G_i. The element x chosen in step 3 is not contained in $\langle L \rangle$, among all remaining elements it has the minimal possible image of β_i (which is a necessary condition for the lexicographically smallest element) and among all such elements x is the smallest by the use of algorithm 7. Thus the computed x is the smallest element not in $\langle L \rangle$ as required. □

The orbit $\beta_i^{G_i}$ in step 2 either contains the orbit of $\beta_{i+1}^{G_{i+1}}$, in which case we need at most as many generators as new points are reached, or is disjoint, in which case there are at most $\left|\beta_i^{G_i}\right| - 1$ new generators.

Furthermore, with each new generator the partially generated subgroup grows. As G has a base of length k, G has size at most n^k, and thus such an extension step can be done at most $\log_2(n^k) = k \log_2 n$ times.

The number of permutation multiplications of algorithm 10 is thus $\mathcal{O}(nk^2 \log n)$ and the time complexity accordingly $\mathcal{O}(n^2 k^2 \log n)$.

4. OTHER ACTIONS

We now turn to the case of a (finite) group G of automorphisms of another algebraic structure A (on which G acts faithfully). Examples are A a vector space and G a matrix group or A a group and G its automorphism group. We assume A to be totally ordered by $<$. We choose a generating sequence B of A and define the ordering $\ll$ on G by lexicographic comparison of the images of B under automorphisms.

REMARK 11. *For the case of a matrix group acting on a vector space we can take B to be the standard basis, then $\ll$ is the natural ordering of matrices mentioned in section 1.*

We now assume that G is finite and denote by Ω the union of the orbits of the elements in B under G: $\Omega = \bigcup_{\beta \in B} \beta^G$. The action of G on Ω is faithful because $A = \langle B \rangle$. We can thus consider G to be a permutation group on Ω and observe that B is a base. By definition of $\ll$ we have for $g, h \in G$ that $g \ll h$ if and only if $B^g < B^h$ lexicographically. (To improve the performance again one can remove those points from B which are stabilized already by the stabilizer of the previous points, however this is not strictly necessary.)

REMARK 12. *It should be pointed out, that we essentially consider the group G as a permutation group. A total ordering of subgroups and cosets of G could be obtained by simply constructing the isomorphic permutation group given by the action on Ω and applying the methods of section 3. However, this ordering would* not *be the lexicographic ordering derived from the natural ordering of the elements described above. The purpose of the algorithms of the current section is to compute that lexicographic ordering.*

In particular, note that $n = |\Omega|$ will be the size of the permutation domain. For matrix groups this is often exponential in the dimension and field size (and so the input length). Complexities will be given in terms of this n.

We now aim to modify the algorithms of the last section to find smallest and largest element with respect to $\ll$ and to find the smallest generating set of G. The problem here is that the base B is not necessarily lexicographically smallest in its G-orbit. (If it is, then the situation is (up to the actual names of the points in Ω) exactly the same as in section 3.) The first step of the calculation will therefore be the calculation of the minimal base image:

ALGORITHM 13. (Smallest Base Image)
Assume that $\{G_i\}_{i=1}^k$ is a stabilizer chain corresponding to $B = (\beta_1, \ldots, \beta_k)$. This algorithm computes an element $min \in G$ such that $C := B^{min}$ is the minimal possible base image as well as a stabilizer chain H_i corresponding to C:

1. [*Initialization*] Let $H_j := G_j$ and $\gamma_j := \beta_j$ for all $j \leq k$, let $C := B$, $i := 1$ and $min := ()$.
2. [*Finished?*] If $H_i = \langle 1 \rangle$ then return min and $\{H_j\}$.
3. [*Make current base point minimal*] Take $rep \in H_i$ such that $\beta_i^{rep} = \min\{\beta_i^s \mid s \in H_i\}$ is the minimal image of the current base point. For $j \geq i$ replace H_j by H_j^{rep} and γ_j by γ_j^{rep}. Let $min := min \cdot rep$. (Now γ_i is the minimal possible image of an i-th base point, the H_j form a stabilizer chain for the new γ_j and g maps the old base B to the base defined by the current γ_i.)
4. [*Loop*] Increment i. Go to step 2.

PROOF. Because we compare base images lexicographically, we can find the minimal base image iteratively in a loop over the base points. □

The element *min* produced by this algorithm maps the base B to the smallest image C, thus it is the smallest element with respect to $\ll$.

In each of the k iterations of the algorithm a (partial) stabilizer chain has to be conjugated. This involves mainly conjugacy of the strong generators. Thus the algorithm requires $\mathcal{O}(ks)$ permutation multiplications, where s is the number of strong generators (and these multiplications dominate the algorithm).

Recall that, to compare subgroups lexicographically, using lemma 6, we need to be able to compute maximum elements, and smallest generating sequences.

If we apply algorithm 7 (in the variation to find the maximal element according to remark 8) to the stabilizer chain H_i, we get an element x which maps C to the largest possible base image L. Therefore $min \cdot x$ maps B to L and thus is the largest element of G with respect to $\ll$.

This leaves the task of computing $\mathcal{S} = \mathcal{S}(G)$. For this we modify algorithm 10. The main difficulty here is that $\mathcal{S}$ is no longer guaranteed to be a strong generating set.

Let x be an element that maps the base C to an image D. Then $min \cdot x$ maps the base B to the same image D. Furthermore elements are smaller (with respect to $\ll$), when they map more initial base points in B to the corresponding points in C (because that yields the lexicographically smallest images). Thus all the elements in the coset $min \cdot H_i$ are smaller than all the remaining elements in $min \cdot H_{i-1}$.

In other words: The smallest generating set $\mathcal{S}$ starts with *min*, then contains elements of $min \cdot H_k$, then of $min \cdot H_{k-1}$ and so on.

Let $\mathcal{S}_i$ be the prefix of $\mathcal{S}$ that contains all the elements in $\mathcal{S} \cap (min \cdot H_i)$.

LEMMA 14. *The orbit orb of γ_i under the stabilizer subgroup $\mathrm{Stab}_{\langle \mathcal{S}_i \rangle}(\gamma_1, \ldots, \gamma_{i-1})$ contains the orbit $\gamma_i^{H_i}$.*

PROOF. Suppose that $\delta \in \gamma_i^{H_i}$ but $\delta \notin orb$. Then there is $h \in H_i$ such that $\gamma_i^h = \delta$. Let $x = min \cdot h \in min \cdot H_i$. By assumption, $h \notin \langle \mathcal{S}_i \rangle$ and thus – as $min \in \mathcal{S}_i$ – $x \notin \langle \mathcal{S}_i \rangle$. But x is smaller than any element not in $min \cdot H_i$. Thus, by the definition of $\mathcal{S}$, x would have to be chosen as generator in $\mathcal{S}$ before any element not in $min \cdot H_i$, which contradicts the definition of $\mathcal{S}$ and $\mathcal{S}_i$. □

LEMMA 15. *Let L be a prefix of $\mathcal{S}$ containing $\mathcal{S}_{i+1}$, $1 \leq i \leq k$ and $K = \mathrm{Stab}_{\langle L \rangle}(\gamma_1, \ldots, \gamma_{i-1})$. Let $x = min \cdot h$ with $h \in H_i$. Then $x \in \langle L \rangle$ if and only if $\gamma_i^h \in \gamma_i^K$. In particular $L \supset \mathcal{S}_i$, if $\gamma_i^K = \gamma_i^{H_i}$.*

PROOF. Since $min \in L$ (it is the first element), we have that $x \in \langle L \rangle$ if and only if $h \in \langle L \rangle$.

Proceed by (reverse) induction: The base case $i = k + 1$ is trivial. Thus assume the statement holds for $i+1$ and we shall prove it for i:
Since L contains $\mathcal{S}_{i+1}$, lemma 14 shows that orbit of γ_{i+1} under $\mathrm{Stab}_{\langle L \rangle}(\gamma_1, \ldots, \gamma_i)$ equals $\gamma_{i+1}^{H_{i+1}}$. Thus, by induction, $\langle L \rangle$ contains every element of H_{i+1}.

If γ_i^h is in the orbit, $K \leq \langle L \rangle \cap H_i$ contains an element in the same H_{i+1}-coset as h, thus $h \in \langle L \rangle$.

Conversely, $h \in \langle L \rangle$ implies that $h \in K$. □

This shows, that we can also find $\mathcal{S}$ in this case in an iterative way, corresponding to the stabilizer chain $\{H_i\}$, by first constructing $\mathcal{S}_k$, then extending to $\mathcal{S}_{k-1}$ and so on:

Consider the step that extends $\mathcal{S}_{i+1}$ to $\mathcal{S}_i$. The elements added will be of the form $min \cdot h$ with $h \in H_i - H_{i+1}$. Therefore they map the first $i-1$ base points $\beta_1, \ldots, \beta_{i-1}$ to $\gamma_1, \ldots, \gamma_{i-1}$ and map β_i to an element of $\gamma_i^{H_i}$. Lemma 15 shows that we have to find elements $h \in H_i$ that give new γ_i images. Furthermore (as *min* maps B to the smallest base C) for $g, h \in H_i$ we have that $g \ll h$ (with respect to the base C) if and only if $min \cdot g \ll min \cdot h$ with respect to B.

This yields the following algorithm, generalizing algorithm 10:

ALGORITHM 16. (Smallest Generating set for arbitrary base) Let *min*, $\{\gamma_i\}_{i=1}^k$ and $\{H_i\}_{i=1}^k$ as computed by algorithm 13 for a group G. The following procedure computes $\mathcal{S}(G)$:

1. [*Initialization*] Let $i := k$ and $L := [min]$.
2. [*Compute stabilizer*] Let $K := \mathrm{Stab}_{\langle L \rangle}([\gamma_1, \ldots, \gamma_{i-1}])$.

3. [*Orbit Reached?*] Let $orb := \gamma_i^{H_i} - \gamma_i^K$. If $orb = \emptyset$ then go to step 5.

4. [*Minimal Element*] Let $\omega := \min(orb)$ and take $t \in H_i$ from the transversal such that $\gamma_i^t = \omega$. Using algorithm 7 compute the minimal element r (with respect to base C) of $H_{i+1}t$. (This uses $H_i > \cdots > H_k$ as a stabilizer chain for H_i.) Add $min \cdot r$ to L. Go to step 2.

5. [*Step*] Let $i := i - 1$. If $i > 0$ then go to step 2. Otherwise return $\mathcal{S}(G) := L$.

PROOF. The element r in step 4 maps γ_i to a new point, fixes all prior base points in C and maps the further points as small as possible. Thus the added generator $max \cdot r$ maps β_i to a new image (and so is a new generator according to Lemma 15), and it maps the remaining base points β_i as small as possible. Thus it is the smallest element not yet contained in $\langle L \rangle$. □

For the complexity, observe that the main difference to algorithm 10 is the additional stabilizer chain calculation in step 2. Assuming that G is a small-base group, and noting that $|G|$ is known, the cost of each of these is $\mathcal{O}(n \log n)$ equivalent to $\mathcal{O}(\log n)$ permutation multiplications, each [5].

This stabilizer computation is performed once for every new generator, of which there are (by the argument after algorithm 10) at most $k \log_2 n$. Thus the total cost for stabilizer computations is $\mathcal{O}\left(k \log^2 n\right)$ permutation multiplications. This is bounded from above by the cost of algorithm 10, the overall cost thus is the same as in the permutation group case, namely $\mathcal{O}\left(n^2 k^2 \log n\right)$. (For a large-base group we get by a similar argument the total cost $\mathcal{O}\left(n^3 k\right)$.)

5. CONCLUDING REMARKS

The algorithms in this paper have been implemented in the system GAP [1] and are used therein for the comparison of subgroups. (The system in general defines the ordering of domains to be based on Definition 1).

In practice the cost of running the algorithms is typically dominated by the initial computation of a stabilizer chain. Thus computations for permutation groups typically are unproblematic. Matrix groups however can have extremely long orbits which can cause memory problems. This limits the applicability of the algorithm to those matrix groups for which the orbits of the vectors in a standard basis are still of reasonably small length.

The limiting factor in practice, however is most frequently the fact that the comparison of subgroups mainly is of interest in computations where a large number of subgroups are to be considered. Creating and storing these subgroups (note that we have to store a stabilizer chain for each subgroup as well) then typically becomes a problem in the first place, even if single pair comparisons still performs well.

The authors would like to thank the referees for their helpful remarks.

6. REFERENCES

[1] The GAP Group, `http://www.gap-system.org`. *GAP – Groups, Algorithms, and Programming, Version 4.3*, 2002.

[2] D. F. Holt. The calculation of the Schur multiplier of a permutation group. In M. D. Atkinson, editor, *Computational Group theory*, pages 307–319. Academic press, 1984.

[3] A. Hulpke. Constructing transitive permutation groups. submitted, `http://www.math.colostate.edu/~hulpke/paper/transgp.html`.

[4] E. M. Luks. Permutation groups and polynomial-time computation. In L. Finkelstein and W. M. Kantor, editors, *Groups and Computation*, volume 11 of *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*, pages 139–175, Providence, RI, 1993. Amer. Math. Soc.

[5] Á. Seress. *Permutation Group Algorithms*. Cambridge University Press, 2003.

[6] C. C. Sims. Determining the conjugacy classes of a permutation group. In G. Birkhoff and M. H. Jr., editors, *Computers in Algebra and Number theory*, pages 191–195, Providence, RI, 1971. Amer. Math. Soc.