Computing Intersections and Normalizers in Soluble Groups

by

S. P. Glasby
Department of Mathematics
Victoria University of Wellington
P.O. Box 600, Wellington, New Zealand

and

Michael. C. Slattery
Dept. of Mathematics, Statistics,
and Computer Science
Marquette University
Milwaukee, WI 53233, U.S.A.

Dedicated to Tim Wall on the occasion of his 65^{th} *birthday.*

Abstract

Let H and K be arbitrary subgroups of a finite soluble group G. The purpose of this paper is to describe algorithms for constructing $H \cap K$ and $N_G(H)$. The first author has previously described algorithms for constructing $H \cap K$ when the indices $|G : H|$ and $|G : K|$ are coprime, and for constructing $N_G(H)$ when $|G : H|$ and $|H|$ are coprime (i.e., when H is a Hall subgroup of G). The intersection and normalizer algorithms described in the present paper are constructed from generalizations of these algorithms and from an orbit-stabilizer algorithm.

1980 *Mathematics Subject Classifications*: 20-04, 20D10.

1. BACKGROUND

Throughout this paper we will assume that G is a finite soluble group defined by a power-commutator presentation

$$G = \langle g_1, \ldots, g_t | g_i^{p_i} = u_{i,i},\ 1 \leq i \leq t,\ [g_j, g_i] = u_{i,j}, 1 \leq i \leq j \leq t \rangle$$

where the $u_{i,j}, 1 \leq i \leq j \leq t$, are words of the form

$$u_{i,j} = g_{i+1}^{k(i,j,i+1)} \ldots g_t^{k(i,j,t)}.$$

(Such a presentation is also called an AG-system [5].) We will further assume that

$$G = \langle g_1, \ldots, g_t \rangle \rhd \langle g_2, \ldots, g_t \rangle \rhd \ldots \rhd \langle g_t \rangle \rhd \langle 1 \rangle \tag{1}$$

is a composition series which refines a normal series

$$G = N_1 \rhd N_2 \rhd \ldots \rhd N_r \rhd N_{r+1} = \langle 1 \rangle$$

for which N_i/N_{i+1} is an elementary abelian q_i-group for $1 \leq i \leq r$. This normal series could be a chief series, however, our algorithms do not assume this. Note that G has composition length t, each p_i is a prime and the sets $\{p_1, \ldots, p_t\}$ and $\{q_1, \ldots, q_r\}$ are equal.

Any non-identity element in G can be represented in the form

$$g_j^{c_j} \, g_{j+1}^{c_{j+1}} \, \ldots \, g_t^{c_t}, \quad 0 \leq c_i < p_i$$

with $c_j \neq 0$. Such an element will be said to have *weight* j and *leading exponent* c_j. We define the weight of the identity element to be $t + 1$.

An *induced sequence of generators* for a subgroup H of G is a sequence $h_1, \ldots, h_m$ such that

$$H = \langle h_1, \ldots, h_m \rangle \rhd \langle h_2, \ldots, h_m \rangle \rhd \ \ldots \ \rhd \langle h_m \rangle \rhd \langle 1 \rangle \tag{2}$$

is a composition series which is obtained from (1) by intersecting with H. More precisely, the above subnormal series is obtained from

$$H = H \cap \langle g_1, \ldots, g_t \rangle \rhd H \cap \langle g_2, \ldots, g_t \rangle \rhd \ldots \rhd H \cap \langle g_t \rangle \rhd H \cap \langle 1 \rangle = \langle 1 \rangle$$

by deleting repeated subgroups. Similarly, if N is a normal subgroup of G, then $x_1 \, N, \ldots, x_m \, N$ is called an *induced sequence of generators* for G/N if

$$G/N = \langle x_1 \, N, \ldots, x_m \, N \rangle \rhd \langle x_2 \, N, \ldots, x_m \, N \rangle \rhd \ldots \rhd \langle x_m \, N \rangle \rhd \langle N \rangle \tag{3}$$

is a composition series which is obtained from (1) by applying the homomorphism $G \to G/N : x \mapsto xN$. In particular, each $x_i \, N$ is the image of some

g_j. Note that if (1) refines a normal series with elementary abelian quotients, then so do (2) and (3).

The following process will be important in several places.

(1.1) ALGORITHM (SIFTING): Let $N \lhd G$ and A be a subgroup of G such that the elements of $AN - N$ have smaller weight than each element of N. Given $x \in AN$ this algorithm will produce $y \in A$ such that $xN = yN$.

Let m be the minimal weight of elements of N.

Let $a_1, a_2, \ldots, a_s$ be an induced generating sequence for A.

$z := x$.

While weight $(z) < m$ do

Find a_i with weight(a_i)=weight(z)

/* this must be possible since $z \in AN$ */.

$c :=$ the leading exponent of z.

$z := a_i^{-c} z$

endwhile.

$y := xz^{-1}$.

The correctness of the sifting algorithm can be seen by noting that z is always in AN and the loop terminates only when $z \in N$.

The main algorithms in this paper involve combining techniques from [2] and [3] with orbit-stabilizer calculations as described in [5]. The orbit algorithm in [5] involves constructing the orbit Δ of an element ω under the action of G by stepping up a composition series in G. One needs to test at each stage whether or not some conjugate of ω is in Δ and if so to identify the element as ω^x for some x in earlier terms in the composition series. One obvious method for producing this x is to store the elements of Δ in an array with each corresponding x stored in a parallel array. While this allows an easy determination of x when needed, it involves the computation of many elements which are never used. John J. Cannon suggested that since the

elements of Δ are computed in a well-defined order, one could reconstruct the appropriate conjugating element based on the position of the element in Δ. This provides a significant savings of time and space.

Since N_i/N_{i+1} is an elementary abelian q_i-group, we can view it as a vector space over the field of q_i elements. We note that the correspondence is a simple matter. If $N_i = \langle g_c, g_{c+1}, \ldots, g_t \rangle$, $N_{i+1} = \langle g_{d+1}, \ldots, g_t \rangle$ and $x \in N_i$ is written

$$x = g_s^{a_c} \ \ldots \ g_t^{a_t}, \qquad 0 \leq a_j < q_i$$

then x corresponds to the vector $(a_c, a_{c+1}, \ldots, a_d)$. It is also easy to move from the vector space back to the factor group since $g_c^{a_c} \, g_{c+1}^{a_{c+1}} \ \ldots \ g_d^{a_d}$ lies in the same coset of N_{i+1} as does x. This shift of viewpoint occasionally allows us to replace group collections with less expensive vector space calculations.

The authors are grateful to John J. Cannon for his helpful discussions. Thanks are also due to the University of Sydney for funding Slattery's visit there, and to Marquette University for the use of their computing facilities. Special thanks to the referees for their many helpful comments.

2. COMPUTING INTERSECTIONS

Let H be a subgroup of G which acts on a set and suppose that an induced sequence of generators is known for H. The orbit-stabilizer algorithm [5] may be used to compute the orbit and stabilizer of a given point. It is noted that paper that intersections may be calculated by finding certain stabilizers. This follows since if K is a subgroup of G, then H acts via right multiplication on the right cosets of K in G, and the stabilizer in H of the right coset K is $H \cap K$. This gives rise to an algorithm for computing intersections which can run slowly when large orbits are encountered. As there is usually no canonical set of right coset representatives of K in G, testing whether Ka lies in the partial orbit $\{Ka_1, \ldots, Ka_m\}$ can involve many element membership tests of the form $a_i \, a^{-1} \in K$.

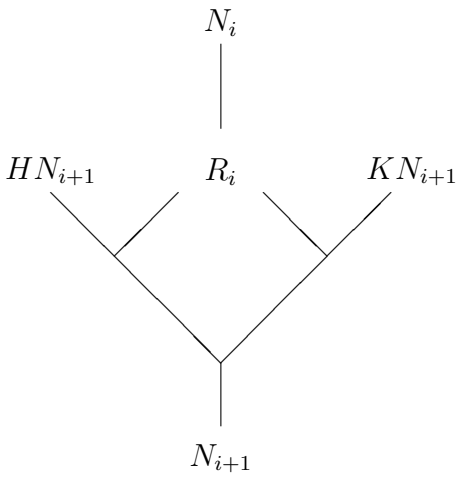
In [3] an algorithm is given for computing $H \cap K$ when $|G : H|$ and $|G : K|$ are coprime. The algorithm computes $HN_{i+1} \cap KN_{i+1}$ from $HN_i \cap KN_i$ for $1 \leq i \leq r$. (Note that $H \cap K = HN_{r+1} \cap KN_{r+1}$.) The calculation of

$HN_{i+1} \cap KN_{i+1}$ from $HN_i \cap KN_i$ is effected by a rewriting process, called the Covering Algorithm, which assumes that H covers N_i/N_{i+1} (i.e., $N_i = HN_{i+1} \cap N_i$), or K covers N_i/N_{i+1}. While this algorithm is very fast, it relies on the stated covering conditions and so is not generally applicable.

The approach of the present paper is to compute $HN_{i+1} \cap KN_{i+1}$ from $HN_i \cap KN_i$ via an intermediate subgroup. Define R_i to be the subgroup

$$R_i = (HN_{i+1} \cap N_i)(KN_{i+1} \cap N_i).$$

We first compute $HR_i \cap KR_i$ from $HN_i \cap KN_i$ using the orbit-stabilizer algorithm, and then use the Generalized Covering Algorithm to calculate $HN_{i+1} \cap KN_{i+1}$ from $HR_i \cap KR_i$ (see diagram).



In the worst case, when $R_i = N_{i+1}$, our algorithm reduces to the orbit reduction algorithm suggested in [5]. On the other hand, if $R_i > N_{i+1}$, then the size of the orbit calculation is reduced. In fact, when $R_i = N_i$, the entire calculation proceeds by element rewriting.

The Generalized Covering Algorithm stated below uses the following well-known vector space algorithm.

(2.1) ALGORITHM (SUM-INTERSECTION): Let W be a finite dimensional vector space, U and V subspaces of W, and $\mathcal{U}$ and $\mathcal{V}$ bases of U and V

respectively. We can compute bases for $U+V$ and $U\cap V$ and a decomposition matrix D as follows.

Construct the matrix $\begin{pmatrix} \mathcal{U} & \mathcal{U} \\ \mathcal{V} & 0 \end{pmatrix}$.

Apply elementary row operations to this matrix to construct the matrix $\begin{pmatrix} \mathcal{P} & \mathcal{Q} \\ 0 & \mathcal{R} \end{pmatrix}$, which is in row echelon form, where the rows of $\mathcal{P}$ are nonzero.

The rows of $\mathcal{P}$ form a basis for $U+V$.

The rows of $\mathcal{R}$ form a basis for $U\cap V$.

Let D be the (non-square) matrix $(\mathcal{P} \quad \mathcal{Q})$.

To see that we get the bases as claimed, note that if (x,y) is a row of a matrix which is row equivalent to $\begin{pmatrix} \mathcal{U} & \mathcal{U} \\ \mathcal{V} & 0 \end{pmatrix}$, then $x \equiv y \pmod{V}$. Hence $R \leq U\cap V$ where R is the row space of $\mathcal{R}$. By dimensions we must have equality.

The decomposition matrix D provides an effective method of realizing elements of $U+V$ as $u+v$ with $u\in U, v\in V$. In particular, if $w\in U+V$ then we can write w as a linear combination of rows in $\mathcal{P}$. By the above comment, we see that if u is the same combination of corresponding rows in $\mathcal{Q}$, then $w\equiv u \pmod{V}$. Thus we can let $v=w-u$.

The calculation of $HN_{i+1}\cap KN_{i+1}$ from $HR_i\cap KR_i$ is accomplished by the following generalization of the Covering Algorithm. Recall that we can view N_i/N_{i+1} as a vector space.

(2.2) ALGORITHM (Generalized Covering): Let $R=R_i$ and $N=N_{i+1}$ for some $i, 1\leq i\leq r$. Let $h_1,\ldots,h_m$ be elements of H and $k_1,\ldots,k_m$ be elements of K such that $h_jR=k_jR, 1\leq j\leq m$, and $h_1R,\ldots,h_mR$ is an induced sequence of generators for $(HR\cap KR)/R$. We construct, as follows, elements $h'_1,\ldots,h'_n$ of H and $k'_1,\ldots,k'_n$ of K such that $h'_jN=k'_jN, 1\leq j\leq n$, and $h'_1N,\ldots,h'_nN$ is an induced sequence of generators for $(HN\cap KN)/N$.

Let W be the vector space corresponding to N_i/N_{i+1} and let U and V be subspaces which correspond to $(HN \cap R)/N$ and $(KN \cap R)/N$ respectively. (Note that induced generators for these quotients, and so bases for U and V, can be identified easily by weight considerations).

Use the Sum-Intersection Algorithm to produce a decomposition matrix for $U + V$ (which corresponds to R/N) and a basis $u_{m+1}, \ldots, u_n$ for $U \cap V$.

for $j = m + 1$ to n do

Let xN be an element of $(H\,N \cap R)/N$ which corresponds to u_j and use the Sifting Algorithm to find $h'_j \in H$ with $h'_j\,N = x\,N$. Since $h'_j\,N (= x\,N)$ corresponds to an element of $U \cap V$, we have $h'_j \in K\,N$ and so we can use the Sifting Algorithm to find $k'_j \in K$ with $k'_j\,N = h'_j\,N$.

endfor.

for $j = 1$ to m do

Let $w \in W$ correspond to $h_j^{-1}\,k_j \in R$ and use the decomposition matrix to write $w = u + v$ with $u \in U, v \in V$.

As above we can compute $x \in H$ and $y \in K$ so that $x\,N$ and $y\,N$ correspond to u and v respectively.

$h'_j := h_j\,x$

$k'_j := k_j\,y^{-1}$

endfor.

In proving the correctness of this algorithm we will need the following lemma.

(2.3) LEMMA: Let H, K, and L be subgroups which normalize R and N. Suppose that $N \subseteq R, LN \subseteq HN \cap KN, LR = HR \cap KR$ and $LN \cap R = HN \cap KN \cap R$. Then $LN = HN \cap KN$.

Proof: We have

$$LR \subseteq (HN \cap KN)\,R \subseteq HR \cap KR = LR.$$

Thus $(LN)R = (HN \cap KN)R$ and $(LN) \cap R = (HN \cap KN) \cap R$ which implies that $LN = HN \cap KN$. $\square$

Proof of Generalized Covering algorithm: We first check that $h'_j\, N = k'_j\, N$ for $1 \leq j \leq n$. If $m < j \leq n$, this is immediate from the construction of the k'_j. For $1 \leq j \leq m$, we see by the definitions of w, x, and y that $h_j^{-1}\, k_j N = xyN$ and so

$$h'_j N = h_j\, xN = k_j\, y^{-1} N = k'_j\, N.$$

Now let $L = \langle h'_1, \ldots, h'_n \rangle$. Then $LN \leq HN \cap KN$. Furthermore, for each j, the elements x and y are in R and so $LR = \langle h'_1 R, \ldots, h'_m R \rangle = \langle h_1 R, \ldots, h_m R \rangle = HR \cap KR$ and $LN \cap R = \langle h'_{m+1} N, \ldots, h'_n N \rangle = (HN \cap R) \cap (KN \cap R)$. Thus by Lemma 2.3, $LN = HN \cap KN$ as claimed. $\square$

Suppose we have two subgroups A and B which satisfy $A \subseteq H, B \subseteq K$, and $AN_i = BN_i$. We now describe an action of A on N_i/R_i. In particular, we claim that A normalizes R_i. This follows from the fact that $HN_i \cap KN_i$ normalizes R_i.

(2.4) DEFINITION: We define an action (depending on B) of A on $N_i\,/R_i$, as follows. Let $a \in A$. Since $AN_i = BN_i$, we can write $a = b_a\, x_a$ with $b_a \in B, x_a \in N_i$. Define

$$(R_i\, x) \cdot a = R_i\, x^a + R_i\, x_a.$$

(Note that this is just an affine map on the vector space N_i/R_i).

The motivation behind this definition lies in the following lemma.

(2.5) LEMMA: The stabilizer in A of the coset R_i under the above action is equal to $A \cap B\ R_i$.

Proof: Consider the action of A on the right cosets of $B\, R_i$ by right multiplication. Since $A \subseteq AN_i = BN_i$, we see that A permutes the cosets of BR_i in BN_i. But $BN_i = (BR_i)\, N_i$ and $BR_i \cap N_i = R_i$, thus we can identify the right cosets of BR_i in BN_i with the elements of N_i/R_i. If $a \in A$, then as above we can write $a = b_a\, x_a$ with $b_a \in B, x_a \in N_i$ and for $x \in N_i$ we have

$$(BR_i\, x)\, a = (Bb_a)\, (b_a^{-1}\, R_i\, b_a)\, x_a\ a^{-1}\, xa = BR_i\, x_a\, (a^{-1}\, xa).$$

One sees that the action in Definition 2.4 is simply this right multiplication action rewritten in terms of the corresponding elements of N_i/R_i. In particular, the coset R_i in N_i/R_i corresponds to the coset BR_i in BN_i and so the stabilizer is simply $A \cap BR_i$. $\square$

A similar observation on the usefulness of affine mappings in computing certain group actions appears in [6].

We are finally ready to state the Intersection Algorithm. The algorithm moves down through the group using an orbit-stabilizer calculation followed by rewriting to compute successive approximations to $H \cap K$.

(2.6) ALGORITHM (INTERSECTION): Let H and K be subgroups of G. This algorithm constructs subgroups $A_i \subseteq H$ and $B_i \subseteq K$ which are shown in Theorem 2.8 to satisfy

$$A_i\, N_i = HN_i \cap KN_i = B_i\, N_i.$$

In particular, we note that $H \cap K = A_{r+1} = B_{r+1}$.

$A_1 := H$.

$B_1 := K$.

for $i = 1$ to r do

Let W be the vector space corresponding to N_i/N_{i+1} and let U and V be subspaces which correspond to $(HN_{i+1} \cap N_i)$ and $(KN_{i+1} \cap N_i)$ respectively.

Use the Sum-Intersection Algorithm to compute $U + V$.

Since $U + V$ corresponds to R_i/N_{i+1}, we can use $W/(U + V)$ to compute the action of A_i on N_i/R_i as described in Definition 2.4. Let C be the stabilizer of R_i.

Let $h_1, \ldots, h_a$ be elements of H such that $h_1\, R_i, \ldots, h_m\, R_i$ is an induced generating sequence for CR_i/R_i.

Use the Sifting Algorithm to find $k_1, \ldots, k_m$ in B_i such that $k_j\, R_i = h_j\, R_i, 1 \leq j \leq m$.

Use the Generalized Covering Algorithm to produce sequences $h'_1, \ldots, h'_n$ and $k'_1, \ldots, k'_n$.

$A_{i+1} := \langle h'_1, \ldots, h'_n \rangle$.

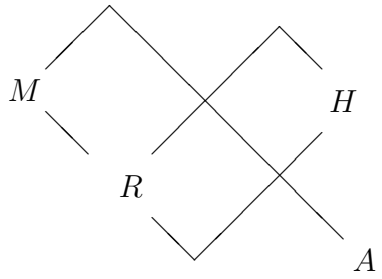
$B_{i+1} := \langle k'_1, \ldots, k'_n \rangle$.

endfor.

We will use the following lemma in the proof of the Intersection Algorithm.

(2.7) LEMMA. If A, B, H, K, R, and M are subgroups satisfying $A \leq H, B \leq K, (H \cap M)(K \cap M) \leq R \leq M$, and $AM = HM \cap KM = BM$, then $HR \cap KR = AR \cap BR$.

Proof: The containment $AR \cap BR \leq HR \cap KR$ follows from $A \leq H$ and $B \leq K$. Conversely, as seen in the diagram



we have

$$\begin{array}{lll} HR \cap KR & \leq HR \cap HM \cap KM & \text{as } R \leq M, \\ & = HR \cap AM & \text{as } AM = HM \cap KM, \\ & = (H \cap AM)\, R & \text{by the modular law}, \\ & = A(H \cap M)\, R & \text{by the modular law}, \\ & = AR & \text{as } H \cap M \leq R. \end{array}$$

A similar argument shows that $HR \cap KR \leq BR$ and hence $HR \cap KR \leq AR \cap BR$. $\square$

We are now ready to prove that the Intersection Algorithm is correct.

(2.8) THEOREM: In Algorithm 2.6, subgroups A_i and B_i satisfy

$$(*) \qquad A_i\, N_i = HN_i \cap KN_i = B_i\, N_i$$

for $1 \leq i \leq r+1$.

Proof: Note that $(*)$ holds for $i = 1$ since $N_1 = G$. Next suppose that $(*)$ is true for $i = l$, where $1 \leq l \leq r$ and consider the main loop of the algorithm.

By Lemma 2.5 we see that $C = A_l \cap B_l R_l$ and so $CR_l = A_l R_l \cap B_l R_l$. However, by $(*)$ and Lemma 2.7 (with $M = N_l$) we have

$$A_l R_l \cap B_l R_l = HR_l \cap KR_l.$$

Thus $h_1 R_l, \ldots, h_a R_l$ is an induced generating sequence for $(HR_l \cap KR_l)/R_l$. Since $CR_l \subseteq BR_l$ we can use the Sifting Algorithm to find $\{k_j\}$ as stated. These sequences $\{h_j\}$ and $\{k_j\}$ satisfy the hypotheses of the Generalized Covering Algorithm. Thus we see that

$$A_{l+1} N_{l+1} = HN_{l+1} \cap KN_{l+1} = B_{l+1} N_{l+1}$$

which is $(*)$ for $i = l + 1$. □

3. IMPLEMENTATION AND PERFORMANCE OF THE INTERSECTION ALGORITHM

Algorithm 2.6 has been implemented in C as part of the computational group theory system CAYLEY [1]. The routine takes as input a power-commutator presentation for G with a specified normal series with elementary abelian factors (see Section 1) and induced generating sequences for subgroups H and K. It returns an induced generating sequence for $H \cap K$.

In the main loop of the Intersection Algorithm we use the Sum-Intersection Algorithm twice, first to compute $U + V$ and then at the beginning of the Generalized Covering Algorithm. In practice, both $U + V$ and $U \cap V$ are computed once and used where needed.

The timings below do not provide a comparison with existing algorithms because no comparable implementation was available. It is expected that

this method will outperform the standard orbit-stabilizer algorithm since the rewriting process is faster and, as explained above, in the worst case we are simply reduced to orbit-stabilizer calculations. Even in that case we benefit from the existence of canonical coset representatives (in N_i / R_i) which allow us to perform the orbit calculations in a vector space. The timings below seem to indicate that the intersection routine is fast enough to be useful.

The six groups listed in Table 3.1 were used to obtain timings for the Intersection and Normalizer Algorithms. Let $\widehat{S_4}$ be a group satisfying $|Z(\widehat{S_4})| = 2, \widehat{S_4}/Z(\widehat{S_4}) \cong S_4$ and $\widehat{S_4} \ncong GL(2,3)$, and let p^{2n+1} denote an extraspecial group of order p^{2n+1}. The split extensions

$$\widehat{S_4} \ltimes (7^{2+1} \ltimes 13^{14+1}) \text{ and } GL(2,3) \ltimes (3^{2+1} \ltimes (2^{6+1} \ltimes 3^{8+1}))$$

are denoted by $2^4 3^1 7^3 13^{15}$ and $2^{11} 3^{13}$ and were constructed using the theory described in [4]. The wreath product S_4 wr S_4 wr S_4 is a permutation group of degree 4^3, and the Borel subgroup $B(n,q)$ of $GL(n,q)$ is the subgroup of upper triangular matrices. Finally, $A\Gamma L(1,q)$ equals $Aut(\mathbb{F}) \ltimes (\mathbb{F}^* \ltimes \mathbb{F}^+)$, where $\mathbb{F}$ is a field with q elements.

Table 3.1

G	$\|G\|$	Normal Series for G
$2^4 3^1 7^3 13^{15}$	$2^4 3^1 7^3 13^{15}$	$2, 3, 2^2, 2, 7^2, 7, 13^{14}, 13$
$2^{11} 3^{13}$	$2^{11} 3^{13}$	$2, 3, 2^2, 2, 3^2, 3, 2^6, 2, 3^8, 3$
S_4 wr S_4 wr S_4	$2^{63} 3^{21}$	$2, 3, 2^2, 2^4, 3^4, 2^8, 2^{16}, 3^{16}, 2^{32}$
$B(4, 2^3)$	$2^{18} 7^4$	$7, 7, 7, 7, 2^3, 2^3, 2^3, 2^3, 2^3, 2^3$
$A\Gamma L(1, 2^{11})$	$2^{11} 11^1 23^1 89^1$	$11, 23, 89, 2^{11}$
$A\Gamma L(1, 3^{10})$	$2^4 3^{10} 5^1 11^2 61^1$	$2, 5, 2, 2, 2, 11, 11, 61, 3^{10}$

The timings in Table 3.2 are measured in CPU seconds and were obtained using CAYLEY V3.6 on an AT&T 3B15 computer.

Table 3.2

G	$\|H\|$	$\|K\|$	$\|H \cap K\|$	time (sec)
$2^4 3^1 7^3 13^{15}$	$2^4 3^1 7^3 13^1$	$2^2 3^1 7^1 13^7$	$2^2 3^1 7^1 13^1$	19
$2^{11} 3^{13}$	2^{11}	$2^3 3^{13}$	2^3	5
S_4 wr S_4 wr S_4	2^{63}	$2^7 3^{21}$	2^7	41
$B(4, 2^3)$	$2^9 7^3$	$2^9 7^3$	$2^3 7^2$	6
$A\Gamma L(1, 2^{11})$	$11^1 23^1 89^1$	$11^1 23^1 89^1$	11	35
$A\Gamma L(1, 3^{10})$	$2^1 3^5 5^1 11^2$	$2^1 3^5 5^1 11^2$	$2^1 11^2$	12

4. COMPUTING NORMALIZERS

Let H be a subgroup of G. In [5] it is noted that the orbit-stabilizer algorithm can be used to compute $N_G(H)$, by considering the conjugation action of G on the conjugates of H. Difficulties can arise when large orbits are encountered. On the other hand, if H is a Hall subgroup, then [2] presents a method for computing $N_G(H)$ which avoids orbit calculations. This method relies on the fact that a Hall subgroup either covers a chief factor of G or is coprime to it. Since this need not be true for an arbitrary subgroup, we cannot simply extend the technique to compute arbitrary normalizers. Instead, we will present a hybrid method in which direct computation of elements is used when possible and an orbit-stabilizer calculation is used otherwise. This proves to be a faster approach than just the orbit-stabilizer algorithm in many situations.

(4.1) LEMMA: Let L and K be normal subgroups of G and assume that U and V are subgroups of G such that

$$\begin{aligned} |U| &= |V| \\ UL &= VL \\ U \cap K &= V \cap K \lhd G \end{aligned}$$

$$\text{and } (|U : U \cap K|, |L|) = 1.$$

Then there is an element x of L such that

$$U^x = V.$$

Proof: Obvious since $U/U \cap K$ and $V/U \cap K$ are Hall subgroups of $UL/U \cap K$. □

In the special case needed for the normalizer calculation below, such an element x can be computed by the following slight variation of the conjugation algorithm of [2] (where a discussion of the formula for x can be found).

(4.2) ALGORITHM: Let U and V be subgroups of G with $|U| = |V|$ and K be a normal subgroup of G contained in $U \cap V$. Assume that $UN_r = VN_r$ and U/K (hence V/K) is a p-group with $p \neq q_r$. Let $u_1 K, \ldots, u_c K$ and $v_1 K, \ldots, v_c K$ be induced generating sequences for U/K and V/K respectively with $u_i KN_r = v_i KN_r$ for $1 \leq i \leq c$. Then we compute $x \in N_r$ with $U^x = V$ as follows:

$x :=$ identity of G.

Find an integer t such that $-tp \equiv 1 \pmod{q_r}$

For $i = c$ downto 1 do

Use the Sifting Algorithm to find $w \in V$ and $z \in N_r$ such that $(u_i^x)^{-1} v_i = wz$.

$x := x \left[z^{v_i} (z^2)^{v_i^2} \ldots (z^{q_r - 1})^{v_i^{q_r - 1}} \right]^t$.

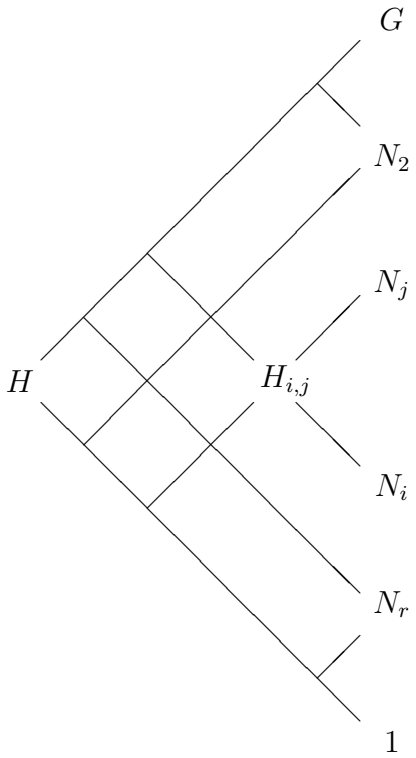
endfor.

The only significant difference between the above algorithm and that in [2] is the use of the echelonization-type sifting process to compute z rather than number theory.

Let H be an arbitrary subgroup of G with $\{N_i\}$ as above and define

$$H_{i,j} = (H \cap N_j)\, N_i, \text{ for } 1 \leq j \leq i \leq r+1.$$

Our approach will be to compute successive normalizers of the subgroups $H_{i,j}$ running over increasing values of i and, for each i, decreasing values of j. Since $H_{1,1} = G$ and $H_{r+1,1} = H$, our scheme will eventually produce the normalizer of H (see diagram).



We now describe an algorithm for computing subgroups $T_{i,j}$ of G. It will be shown below (Theorem 4.5) that $T_{i,j} = N_G(H_{i-1,1}) \cap N_G(H_{i,j})$. Note that this is simply the intersection of the normalizers of all subgroups prior to $H_{i,j}$ in the ordering mentioned above.

(4.3) ALGORITHM (NORMALIZER): We define $T_{i,j} \subseteq G$ as follows:

$T_{1,1} := G$.

For $i = 2$ to $r + 1$ do

$T_{i,i} := T_{i-1,1}$.

For $j = i - 1$ downto 1 do

If q_{i-1} is equal to q_j

Use the orbit-stabilizer algorithm to compute the stabilizer S of $H_{i,j}$ under the action of $T_{i,j+1}$ (acting by conjugation in G).

$T_{i,j} := S$

else

Let $x_1, \ldots, x_s$ be an induced generating sequence for $T = T_{i,j+1}$.

For each x_k, use Algorithm 4.2 to compute $y_k \in T \cap N_{i-1}$, such that

$$(H_{i,j}^{x_k})^{y_k} = H_{i,j}.$$

$T_{i,j} := \langle x_1 y_1, \ldots, x_s y_s, N_{T \cap N_{i-1}}(H_{i,j}) \rangle$.

endif.

endfor.

endfor.

Notes:

(*i*) In the last case above, $N_{T \cap N_{i-1}}(H_{i,j})$ is just the centralizer of $H_{i,j}$ in $T \cap N_{i-1}$ modulo $H_{i,j} \cap N_{i-1}$ and so can be computed as a fixed-point subgroup using linear algebra (see Section 5).

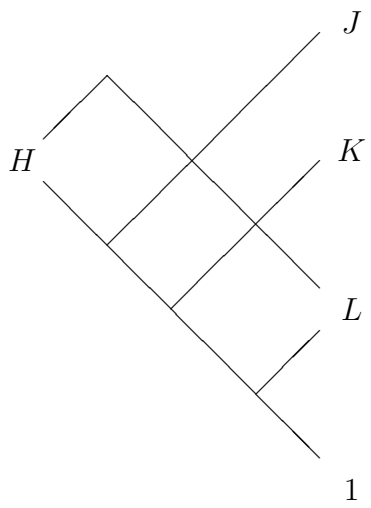
(*ii*) The calculation of $T_{i,j}$ can in fact be carried out modulo N_i thus reducing the size of the group in which you are working.

The proof that the above algorithm computes $N_G(H)$ will require the following factorization lemma.

(4.4) LEMMA: Let $L \subseteq K \subseteq J$ be normal subgroups of G such that $(|J:K|, |L|) = 1$ and let $H \subseteq G$ with $HL \lhd G$. Then

$$N_G(H \cap K) = N_G(H \cap J)\,[N_G(H \cap K) \cap L]\,.$$

Proof: (see diagram)



Let $T = N_G(H \cap K)$. Then we note that

$$HL \cap J \cap T \lhd T$$

and

$$H \cap J \subseteq HL \cap J \cap T.$$

Furthermore $|H \cap J : H \cap J \cap K| = |H \cap J : H \cap K|$ divides $|J:K|$ and so is coprime to $|L|$.

Now, taking $HL \cap J \cap T$ for G in Lemma 4.1 (with $L \cap T$ and $HL \cap K \cap T$ as the normal subgroups), we see that any two T conjugates of $H \cap J$ are conjugate in $HL \cap J \cap T$. Thus we can use the Frattini argument to conclude that

$$T = (HL \cap J \cap T)\; N_T\,(H \cap J).$$

But now

$$N_G(H \cap J) \subseteq N_G(H \cap K) = T$$

and

$$HL \cap J \cap T = (H \cap J)L \cap T = (H \cap J)(T \cap L)$$

thus the above can be rewritten

$$T = N_G(H \cap J)(T \cap L)$$

as claimed. □

In the following statement, let $H_{0,1}$ denote G.

(4.5) THEOREM: $T_{i,j} = N_G(H_{i-1,1}) \cap N_G(H_{i,j})$.

Proof: We will proceed by induction on the subscripts of $T_{i,j}$ in the order (1,1), (2,2), (2,1), (3,3), (3,2), ... Clearly $T_{1,1}\,(= G)$ is the normalizer of $H_{0,1}$ and $H_{1,1}$ (each $= G$). If $i > 1$ and $j = i$, then $T_{i,i} = T_{i-1,1}$. By the inductive hypothesis

$$\begin{aligned} T_{i,i} = T_{i-1,1} &= N_G(H_{i-2,1}) \cap N_G(H_{i-1,1}) \\ &= N_G(HN_{i-2}) \cap N_G(HN_{i-1}) \\ &= N_G(HN_{i-1}) \\ &= N_G(H_{i-1,1}) \end{aligned}$$

since $HN_{i-2} = (HN_{i-1})N_{i-2}$.
However

$$H_{i,i} = N_i \lhd G$$

and so

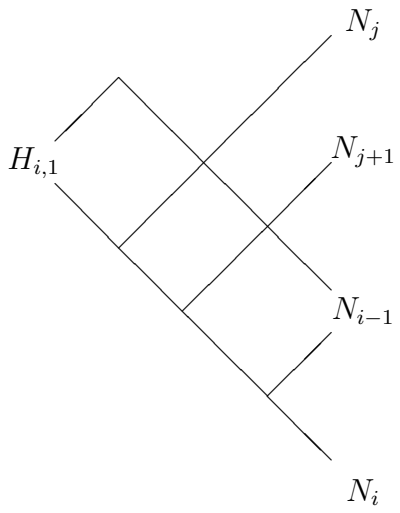
$$T_{i,i} = N_G(H_{i-1,1}) \cap N_G(H_{i,i}).$$

Finally, if $i > 1$ and $j < i$, then we have two cases.

Case $q_{i-1} = q_j$: In this case $T_{i,j}$ is defined to be $N_{T_{i,j+1}}(H_{i,j})$. Since $N_G(H_{i,j}) \subseteq N_G(H_{i,j+1})$ we have

$$\begin{aligned} T_{i,j} &= N_{T_{i,j+1}}(H_{i,j}) = T_{i,j+1} \cap N_G(H_{i,j}) \\ &= N_G(H_{i-1,1}) \cap N_G(H_{i,j+1}) \cap N_G(H_{i,j}) \\ &= N_G(H_{i-1,1}) \cap N_G(H_{i,j}). \end{aligned}$$

Case $q_i - 1 \neq q_j$: Let $R = N_G(H_{i-1,1}) \cap N_G(H_{i,j})$ and let $M = T_{i,j+1} \cap N_{i-1}$. Now let $G^* = N_{G/N_i}(H_{i-1,1})$ play the role of G in Lemma 4.4. Identifying the following Hasse diagram



with the one for Lemma 4.4 shows that

$$N_{G^*}(H_{i,j+1}) = N_{G^*}(H_{i,j})[N_{G^*}(H_{i,j+1}) \cap (N_{i-1}/N_i)].$$

However,

$$N_{G^*}(H_{i,j+1}) = (N_G(H_{i-1,1}) \cap N_G(H_{i,j+1}))/N_i = T_{i,j+1}/N_i$$

and so

$$T_{i,j+1} = RM.$$

On the other hand, the construction of $T_{i,j}$ only modifies the generators of $T_{i,j+1}$ by elements of M and so

$$T_{i,j+1} = T_{i,j}\, M.$$

Since $T_{i,j} \subseteq R$, we need only show that $T_{i,j} \cap M = R \cap M$ in order to conclude that $T_{i,j} = R$. However,

$$M \subseteq N_{i-1} \subseteq N_G(H_{i-1,1})$$

and so

$$\begin{aligned} R \cap M &= N_G(H_{i,j}) \cap M \quad \text{as } M \leq N_{i-1} \leq N_G(H_{i-1,1}) \\ &= N_M(H_{i,j}) \\ &= T_{i,j} \cap M \qquad \text{from the algorithm.} \end{aligned}$$

Thus

$$T_{i,j} = R = N_G(H_{i-1,1}) \cap N_G(H_{i,j}). \qquad \square$$

In particular, Theorem 4.5 tells us that $T_{r+1,1} = N_G(H)$ and so, as mentioned above, this normalizer can be computed by Algorithm 4.3.

5. IMPLEMENTATION AND PERFORMANCE OF THE NORMALIZER ALGORITHM

Algorithm 4.3 has been implemented in C as part of the computational group theory system CAYLEY. The routine takes as input a pc-presentation for G with a specified normal series with elementary abelian factors (see Section 1) and an induced generating sequence for a subgroup H and returns an induced generating sequence for $N_G(H)$.

The routine proceeds by computing a sequence of elements of G which generate $T_{i,j}$ modulo N_i rather than computing a full generating sequence for $T_{i,j}$. In addition to the calculations described in Algorithm 4.3, there are some special cases which allow us to speed up the routine. Since G/N_2 is abelian, we may take all of the generators of G which are outside of N_2 as our initial sequence (corresponding to $T_{2,1}$). When we reach the first step in the outer loop ($T_{i,i} = T_{i-1,1}$) we copy those generators of N_{i-1} which are not in N_i into our sequence to reflect the fact that $T_{i,i}$ must be considered modulo N_i whereas $T_{i-1,1}$, was computed modulo N_{i-1}. If $N_{i-1} \subseteq HN_i$, then $H_{i,1} = H_{i-1,1}$ and so our sequence generates $T_{i,1}$ and we continue with the next iteration of the main loop. Similarly if $H \subseteq N_i$, then $T_{i,1} = T_{i,i}$ and we move to the next iteration.

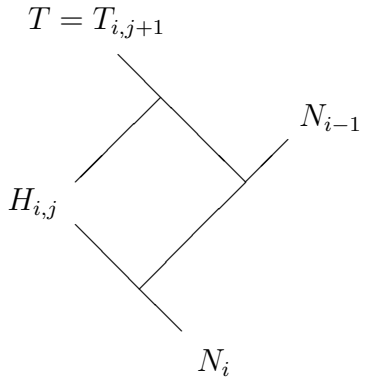
When performing the orbit-stabilizer calculation (q_{i-1} equal to q_j) one is required to determine whether a particular subgroup occurs in a long list of

subgroups (the orbit) and, if so, to determine its position. This is accomplished very quickly by means of a hash table as follows.

Given a subgroup U of a group G, we form a bit string (hash key) based on the canonical generating system of U with respect to the given presentation of G. This canonical generating sequence (see [5] for a more formal treatment) is an induced generating sequence such that the leading exponent of each element is 1 and the corresponding exponents of the other elements have been reduced to zero by cancellation. This produces a sequence of "echelonized" generators for U which are uniquely determined by U (and the presentation of G). By listing the number of generators of U followed by the exponent vectors of these generators as digits in a long integer, we produce a string of bits which uniquely determines the subgroup U. This key can be assigned a location in an appropriately sized array by standard hashing techniques. To check whether a given subgroup occurs in the table, one computes the canonical generating sequence, reads off the hash key, and does a standard hash table lookup. This seems to provide a rather fast and efficient method of dealing with sets of subgroups of a given group.

In the particular setting of the normalizer algorithm, the subgroups in our set are all conjugates of $H_{i,j}$ under the action of $T_{i,j+1}$. Thus we know that portions of the canonical generating sequences will be identical. For instance, the subgroups have the same number of generators, they are the same modulo N_{i-1}, etc. We take advantage of these features to shorten the bit string actually used for the hash key.

In the case where q_{i-1} is not equal to q_j, we need to produce an induced generating sequence for $N_{T\cap N_{i-1}}(H_{i,j})$(see diagram).



To find such a sequence, first note that if we work modulo $H_{i,j} \cap (T \cap N_{i-1})$ which is equal to $H_{i,j} \cap N_{i-1}$, then the desired normalizer is just the centralizer of $H_{i,j}$ in $T \cap N_{i-1}$. Since $(T \cap N_{i-1})/(H_{i,j} \cap N_{i-1})$ is an elementary abelian q_{i-1}-group, we can view this centralizer as the fixed-point subgroup under the action of $H_{i,j}$. Computing the matrices corresponding to the action of the generators of $H_{i,j}$ on $T \cap N_{i-1}$ (modulo $H_{i,j} \cap N_{i-1}$) reduces this calculation to a series of null space computations (i.e. M fixes x if and only if $(M - I)$ annihilates x). We note finally that the generators of $H_{i,j} \cap N_{i-1}$ are easily identified by weight and so we can construct the appropriate matrices directly from the action of $H_{i,j}$ on $T \cap N_{i-1}$ in G (without needing to compute the quotient group). In this way a generating sequence for $N_{T \cap N_{i-1}}(H_{i,j})$ is produced by a series of easy linear algebra calculations.

In Table 5.1 we compare execution times for the implementation of Algorithm 4.3 and a comparable implementation of the standard orbit-stabilizer algorithm (using orbit-reduction via the normal series). These times were obtained running CAYLEY V3.6 on an AT&T 3B15 computer at Marquette University. (A detailed description of the groups can be found in Section 3).

Table 5.1 compares the performance of three algorithms: (1) the Normalizer Algorithm, (2) the algorithm that constructs successive $T_{i,j}$ by using the orbit-stabilizer algorithm, and (3) the algorithm that constructs successive $N_G(H_{1,j})$ by using the orbit-stabilizer algorithm. Let t_1, t_2 and t_3 denote the respective times, in CPU seconds, for these algorithms to compute $N_G(H)$. The entries > 3000 indicate the routine completed in more than 3000 CPU seconds, while the entries $**$ indicate that the routines did not complete.

Table 5.1

G	$\lvert H\rvert$	$\lvert N_G(H)\rvert$	t_1(sec)	t_2(sec)	t_3(sec)
$2^4 3^1 7^3 13^{15}$	$2^4 3^1 7^3 13^1$	$2^4 3^1 7^3 13^1$	21	> 3000	> 3000
$2^{11} 3^{13}$	2^{11}	2^{11}	58	435	2786
S_4 wr S_4 wr S_4	3^{21}	$2^7 3^{21}$	930	> 3000	> 3000
$B(4, 2^3)$	$2^9 7^3$	$2^9 7^4$	64	71	27
$A\Gamma L(1, 2^{11})$	$11^1 23^1 89^1$	$11^1 23^1 89^1$	6	73	137
$A\Gamma L(1, 3^{10})$	$2^1 3^5 5^1 11^2$	$2^2 3^5 5^1 11^2$	44	**	**

References

[1] J. J. Cannon, *An introduction to the group theory language, Cayley*, in "Proceedings of the *London Math. Soc. Symp.* on Computational Group Theory, Durham 1982", Edited by M. D. Atkinson, Academic Press, 1984, pp. 148–183.

[2] S. P. Glasby, *Constructing normalisers in finite soluble groups*, *J. Symbolic Computation* **5** (1988), 285–294.

[3] S. P. Glasby, *Intersecting subgroups of finite soluble groups*, *J. Symbolic Computation* **5** (1988), 295–301.

[4] S. P. Glasby and R. B. Howlett, *Extraspecial towers and Weil Representations*, in preparation.

[5] R. Laue, J. Neubüser and U. Schoenwaelder, *Algorithms for finite soluble groups and the SOGOS system*, in "Proceedings of the *London Math. Soc. Symp.* on Computational Group Theory, Durham 1982", Edited by M. D. Atkinson, Academic Press, 1984, pp. 105–135.

[6] M. Mecky and J. Neubüser, *Some remarks on the computation of conjugacy classes of soluble groups*, *Bull. Austral. Math. Soc.* **40** (1989), 281–292.