J. Symbolic Computation (1988) **5**, 285–294

Constructing Normalisers in Finite Soluble Groups

S. P. GLASBY[†]

Department of Pure Mathematics, The University of Sydney, New South Wales 2006, *Australia*

(*Received* 1 *November* 1985)

This paper describes algorithms for constructing a Hall π-subgroup H of a finite soluble group G and the normaliser $N_G(H)$. If G has composition length n, then H and $N_G(H)$ can be constructed using $O(n^4 \log |G|)$ and $O(n^5 \log |G|)$ group multiplications, respectively. These algorithms may be used to construct other important subgroups such as Carter subgroups, system normalisers and relative system normalisers. Computer implementations of these algorithms can compute a Sylow 3-subgroup of a group with $n = 84$, and its normaliser in 47 seconds and 30 seconds, respectively. Constructing normalisers of arbitrary subgroups of a finite soluble group can be complicated. This is shown by an example where constructing a normaliser is equivalent to constructing a discrete logarithm in a finite field. However, there are no known polynomial algorithms for constructing discrete logarithms.

1. Introduction and Basic Ideas

The normalisers of Hall π-subgroups play an important role in the theory of finite soluble groups, occurring in the Frattini argument and in Burnside's lemma (Hall, 1959, Lemma 14.3.1). They also occur in the definitions of many important subgroups such as Carter subgroups and system normalisers, as well as providing a rich source of abnormal subgroups. This paper describes algorithms for constructing Hall π-subgroups and their normalisers.

The groups discussed in the present paper will be finite and soluble. For reasons mentioned later we choose to represent these groups by *power-commutator* (or *pc*-) *presentations*. A *pc*-presentation is a presentation of the form

$$G = \langle g_1, \ldots, g_n \mid g_i^{p_i} = u_{ii},\ 1 \leqslant i \leqslant n,\ [g_j, g_i] = u_{ij},\ 1 \leqslant i < j \leqslant n \rangle, \tag{1}$$

in which

(a) u_{ij}, $1 \leqslant i \leqslant j \leqslant n$, is a word of the form $g_{i+1}^{k(i,j,i+1)} \ldots g_n^{k(i,j,n)}$ with $0 \leqslant k(i,j,l) < p_l$,
(b) each p_i is prime,
(c) $|G| = p_1 \cdots p_n$, and
(d) there exists a normal series $G = N_1 \rhd \ldots \rhd N_{r+1} = \langle 1 \rangle$, where each N_i/N_{i+1} is elementary abelian and each N_i has the form $\langle g_{n(i)}, g_{n(i)+1}, \ldots, g_n \rangle$ for some integer $n(i)$.

Every finite soluble group is isomorphic to a *pc*-presentation and conversely, every group defined by a *pc*-presentation is finite and soluble. When G is a p-group it is customary to choose the normal series so that each N_i/N_{i+1} has order p. In this case

† This research was undertaken at Rutgers University, N.J., U.S.A. and was supported by a Commonwealth Postgraduate Research Award.

$[g_j, g_i]$, $1 \leqslant i < j \leqslant n$, is a word in the generators $g_{j+1}, \ldots, g_n$. The presentations of the form (1) which satisfy (a), (b), (c), but possibly not (d), have been called AG-systems by Jürgensen (1970), from the German Auflösbare Gruppe.

An example of a *pc*-presentation is given for the symmetric group S_4 of order 24. Let

$$G = \langle g_1, g_2, g_3, g_4 \mid g_1^2 = g_3,\ g_2^3 = g_3^2 = g_4^2 = 1,\ [g_2, g_1] = g_2 g_3,\ [g_3, g_1] = 1,\ [g_4, g_1] = g_3,\ [g_3, g_2] = g_3 g_4,\ [g_4, g_2] = g_3,\ [g_4, g_3] = 1 \rangle. \quad (2)$$

Then identifying g_1, g_2, g_3, g_4 with the permutations $(1, 2, 3, 4)$, $(1, 2, 3)$, $(1, 3)(2, 4)$, $(1, 2)(3, 4)$ establishes an isomorphism between G and S_4. The presentation (2) shows that the subgroups $N_2 = \langle g_2, g_3, g_4 \rangle$, $N_3 = \langle g_3, g_4 \rangle$ and $N_4 = \langle 1 \rangle$ are normal in G. The subgroup N_2 is the alternating group and N_3 is the Klein 4-group.

C. R. Leedham-Green (personal communication, 1983) has developed an algorithm which takes as input a soluble permutation group and outputs a *pc*-presentation together with an isomorphism between the two presentations. If G is a soluble permutation group of degree d, then this algorithm is polynomial in d. Indeed, there is a connection between the complexity of permutation group algorithms and the complexity of algorithms for groups endowed with *pc*-presentations. If G has degree d and composition length n, then $2^n \leqslant d! \leqslant d^d$ and hence $n \leqslant d \log_2 d \leqslant d^2$. Therefore any algorithm which is polynomial in n may be regarded as a permutation group algorithm which is polynomial in d. The converse, however, is false because the minimal degree of a permutation group of composition length n may be exponential in n, as shown by the cyclic group C_{p^n}.

Perhaps one of the greatest virtues of *pc*-presentations, compared to matrix and permutation representations, is that arbitrary quotient groups are easily represented. The quotient groups in this paper all have the form G/N_i. Note that the *pc*-presentation

$$G/N_3 = \langle g_1, g_2 \mid g_1^2 = g_2^3 = 1, [g_2, g_1] = g_2 \rangle$$

may be obtained from (2) by setting $g_3 = g_4 = 1$. This quotient property is vitally important to the normaliser algorithm which recursively constructs $N_G(H)$ from $N_{G/N_r}(HN_r/N_r)$.

If G is defined by the *pc*-presentation (1), then every element of G can be uniquely represented as a *collected* word $g_1^{k_1} \ldots g_n^{k_n}$, with $0 \leqslant k_i < p_i$. The expressions u_{ii} and u_{ij} for $g_i^{p_i}$ and $[g_j, g_i]$ can be used to rewrite the product, $(g_1^{k_1} \ldots g_n^{k_n})(g_1^{l_1} \ldots g_n^{l_n})$; of two collected words as a collected word, $g_1^{m_1} \ldots g_n^{m_n}$. This rewriting process was called *collection* by Hall (1969, p. 29), and is well suited to implementation on a computer, see Felsch (1976) and the references therein.

Denote by G_i the subgroup $\langle g_i, \ldots, g_n \rangle$, for $1 \leqslant i \leqslant n$. An element $g \in G_i$ is said to have *weight* i, written $w(g) = i$, if $g \notin G_{i+1}$. By convention G_{n+1} is the trivial subgroup, and the identity element has weight $n+1$. Let H be a subgroup of G, then the standard composition series $G = G_1 \rhd \ldots \rhd G_{n+1} = \langle 1 \rangle$ for G induces a composition series $H = H_1 \rhd \ldots \rhd H_{m+1} = \langle 1 \rangle$ for H, where $H_i = H \cap G_{j(i)}$. Furthermore, this composition series refines the normal series $H = H \cap N_1 \trianglerighteq \ldots \trianglerighteq H \cap N_{r+1} = \langle 1 \rangle$, which has elementary abelian factors. A sequence $h_1, \ldots, h_m$ of elements of H may be chosen so that $H_i/H_{i+1} = \langle h_i H_{i+1} \rangle$, and hence $w(h_1) < \ldots < w(h_m) < n+1$. By replacing each h_i by a power of itself, if necessary, we may additionally assume that $h_i G_{j+1} = g_j G_{j+1}$, where $j = w(h_i)$. A sequence $h_1, \ldots, h_m$ which satisfies these properties is called an *induced sequence of pc-generators* for H (relative to G), or more briefly, an *induced sequence* for H.

Let G be a group which is defined by a *pc*-presentation and acts on a set Ω. If $\alpha \in \Omega$, then an algorithm due to C. R. Leedham-Green (Laue *et al.*, 1984, p. 110) may be used to calculate the orbit α^G, and the stabiliser G_α. As G acts via conjugation on the set of conjugates of H, the normaliser $N_G(H)$, may be calculated as the stabiliser in G of H. A more effective algorithm for constructing $N_G(H)$ is to recursively construct the stabiliser $N_G(HN_{i+1}/N_{i+1})$, from $N_G(HN_i/N_i)$. However, when the factors N_i/N_{i+1} are large, these stabilisers can be costly to compute.

2. Theorems and Examples

The following lemmas and theorems are basic to the ensuing algorithms. The proofs of Lemmas 1, 2 and 3 are not difficult and the proofs of Theorems 4, 5 and 6 appear in Hall (1959) on pages 141, 46 and 155, respectively.

LEMMA 1. *Let N be an abelian normal subgroup of G. If $h \in G$, then the map $[h, -]: N \to N: x \mapsto [h, x]$, satisfies $[h, xy] = [h, x][h, y]$. If N is elementary abelian, then $[h, -]$ may be regarded as a linear transformation of a vector space.*

LEMMA 2. *Let $G = HN$ be a split extension of N by H. Then $N_G(H) = H \times C_N(H)$.*

LEMMA 3. *Let N be a normal π-subgroup of G. If H/N is a π-subgroup (respectively Hall π-subgroup) of G/N, then H is a π-subgroup (respectively Hall π-subgroup) of G.*

THEOREM 4. *Let G be a finite soluble group and let π be a set of primes. Then G contains a Hall π-subgroup, any two Hall π-subgroups are conjugate, and each π-subgroup of G is contained in a Hall π-subgroup of G.*

THEOREM 5. *If H is a Hall π-subgroup of G, then every subgroup containing $N_G(H)$ is self-normalising in G.*

THEOREM 6. *If each Sylow p-subgroup of the finite group G is normal, then G is nilpotent.*

The normalisers of two cyclic groups are constructed to motivate the ensuing discussions. The first example shows how to construct certain centralisers, and the second, how to conjugate certain subgroups. Let G be the symmetric group of order 24 defined by (2).

For the first example, let $h = g_1g_2g_4$ generate the subgroup H of order 2. Then $N_G(H) = C_G(h)$. The idea is to inductively construct an induced sequence for $N_G(HN_{i+1})$ from an induced sequence for $N_G(HN_i)$. Note that $N_G(HN_i)$ is the pre-image in G of $N_{G/N_i}(HN_i/N_i)$, so that the calculations may be carried out in the smaller group G/N_i.

Now hN_2 is an induced sequence for $N_{G/N_2}(HN_2/N_2)$. (Here we have used the obvious definition for *weight* in G/N_2.) If $\bar{\ }$ denotes the natural homomorphism $G \to G/N_3$, then using Lemma 2, $N_{\bar{G}}(\bar{H}) = \bar{H} \times C_{\bar{N}_2}(\bar{H})$. However, $\overline{[g_2, h]} = \overline{g_2}$ so that $C_{\bar{N}_2}(\bar{H})$ is trivial and $\bar{h}$ is an induced sequence for $N_{\bar{G}}(\bar{H})$. Applying Lemma 2 to HN_3 gives $N_{HN_3}(H) = H \times C_{N_3}(H)$. However, by Lemma 1 $[h, -]$ induces a linear transformation of N_3, and so the kernel of $[h, -]$ is $C_{N_3}(H)$. Since $[h, g_3] = [h, g_4] = g_3g_4$, it follows that $[h, g_3g_4] = 1$ and hence that $g_1g_2g_4, g_3g_4$ is an induced sequence for $N_G(H)$.

For the second example, let $h = g_2$ and $H = \langle h \rangle$. Then $\bar{g}_1, \bar{h}$ is an induced sequence for $N_{\bar{G}}(\bar{H})$. By Lemma 2, $N_{HN_3}(H) = H \times C_{N_3}(H)$. However, the equations $[h, g_3] = g_3g_4$ and $[h, g_4] = g_3$ together with Lemma 1, imply that $C_{N_3}(H)$ is trivial. By Theorem 4, there is an element $x \in N_3$ such that $H^{g_1} = H^x$. Therefore, $g_1 x^{-1}, h$ is an induced sequence for $N_G(H)$. Now $h^{g_1} = g_2^2 g_3$, so let $k = (h^{g_1})^2 = g_2 g_4$, then $h^{-1}k = g_4$. The equation $[h, x] = g_4$ can be solved for $x \in N_3$. In this case, there is precisely one solution, viz. $x = g_3 g_4$. Premultiplying the equation $[h, x] = h^{-1}k$ by h gives $h^x = k$, or $H^x = H^{g_1}$. Hence, $g_1 g_3 g_4, g_2$ is an induced sequence for $N_G(H)$.

3. Preliminary Algorithms

The sifting, centraliser, conjugation and Hall π-subgroup algorithms will be discussed in this section. The sifting algorithm may be used to write an element of a subgroup as a collected work in an induced sequence of *pc*-generators for the subgroup. The centraliser algorithm constructs certain centralisers, and the conjugation algorithm is used to construct a conjugating element given two Hall π-subgroups. The conjugation algorithm is used by the Hall π-subgroup algorithm to construct a Hall π-subgroup. The centraliser and conjugation algorithms are motivated by the techniques of the previous section.

THE SIFTING ALGORITHM

This algorithm is used by an algorithm for constructing an induced sequence of *pc*-generators given an arbitrary set of generators for a subgroup. The algorithm is due to M. F. Newman (Laue *et al.*, 1984, p. 108) and is called the non-commutative Gauss algorithm or the echelonisation algorithm.

Input: Elements $y, k_1, \ldots, k_m$ of G where $w(k_1) < \ldots < w(k_m)$ and for each i, $k_i G_{j+1} = g_j G_{j+1}$ where $j = w(k_i)$.

Output: An element z of G such that yz^{-1} equals a "collected" word $k_1^{c_1} \ldots k_i^{c_i}$ where $c_i \neq 0$ and i is as large as possible. (The word is said to be collected if for $1 \leqslant l \leqslant i$, $0 \leqslant c_l < p_j$ where $j = w(k_l)$.)

```
(1)  Set $z = y$.
(2)  If $w(z) = n+1$, or $w(k_i) \neq w(z)$ for all $i$, then stop.
(3)  Suppose that $w(k_i) = w(z) = s$, say.
(4)  If $zG_{s+1} = g_s^{c_i} G_{s+1}$, then set $z = k_i^{-c_i} z$ and go to step 2. (Note that
     $w(k_i^{-c_i} z) > w(z)$.)
```

This procedure is called *sifting y through* $k_1, \ldots, k_m$. If $k_1, \ldots, k_m$ is an induced sequence for a subgroup K of G, and $y \in K$, then y may be written as a collected word in the generators by sifting y through $k_1, \ldots, k_m$. In this case z equals the identity element.

THE CENTRALISER ALGORITHM

In order to calculate the normaliser in Lemma 2, we shall calculate the centraliser $C_N(H)$ where H normalises N. When N is elementary abelian, Lemma 1 reduces the calculation of $C_N(H)$ to a calculation in linear algebra, or to be more precise, to an application of the Gaussian elimination algorithm. Recall that N_r is a non-trivial elementary abelian normal subgroup of G which is defined by the *pc*-presentation of G.

Input: A subgroup H which normalises the subgroup N_r of G. Let $h_1, \ldots, h_a$ be an induced sequence for H and let $g_m, \ldots, g_n$ be an induced sequence for N_r.

Output: An induced sequence $z_1, \ldots, z_c$ for $C_{N_r}(H)$.

(1) If $a > 1$, then recursively construct $M = C_{N_r}(\langle h_2, \ldots, h_a \rangle)$. Then h_1 normalises both N_r and $\langle h_2, \ldots, h_a \rangle$ and thus M. Recursively construct $C_M(\langle h_1 \rangle)$, then set $C_{N_r}(H) = C_M(\langle h_1 \rangle)$ and stop.

(2) Suppose now that $a = 1$. For $m \leqslant i \leqslant n$ find the collected word $g_m^{b_{im}} \ldots g_n^{b_{in}}$ for $[h_1, g_i]$. Then the linear transformation $[h_1, -]$ of N_r may be represented by the $(n-m+1) \times (n-m+1)$ matrix $B = (b_{ij})$ with respect to the basis $g_m, \ldots, g_n$ for N_r. A basis for the kernel of B may be determined by applying elementary row operations to the augmented matrix $[B \mid I]$. Let $z_1, \ldots, z_c$ be the elements of N_r corresponding to the basis elements of the kernel.

THE CONJUGATION ALGORITHM

Let H and K be two Hall π-subgroups of G which are equal modulo N_r. The conjugation algorithm constructs an element $g \in N_r$ which conjugates H to K. The conjugation algorithm is useful for constructing Hall π-subgroups and their normalisers. Kantor & Taylor (1987) have described a different algorithm which finds a conjugating element given two Sylow subgroups of a permutation group.

Input: Two Hall π-subgroups H and K of G which are equal modulo N_r.
Output: An element $x \in N_r$ such that $H^x = K$.

(1) If N_r is a π-subgroup, then $H = K$ so set $x = 1$ and stop.

(2) Henceforth, assume that N_r is an elementary abelian π'-subgroup. If K has composition length greater than one, then let $L = K \cap G_j$ be a normal subgroup of K of prime index p where G_j is the subnormal subgroup $\langle g_j, \ldots, g_n \rangle$ of G and where $HN_r \leqslant G_{j-1}$. Since $H \cap G_j$ and L are Hall π-subgroups of G_j, recursively find $y \in N_r$ such that $(H \cap G_j)^y = L$.

(3) Let $H^y = \langle h, L \rangle$ and $K = \langle k, L \rangle$ where $hG_j = kG_j = g_{j-1}G_j$. Factorise the order of $h^{-1}k$ into a π-part s, and a π'-part s'. (The order $|g|$, of $g \in G$ may be recursively factorised into primes as follows. If $w(g) = n+1$, then $|g| = 1$, otherwise $|g| = p_k |g^{p_k}|$ where $k = w(g)$.) If $s' = 1$, then set $x = y$ and stop.

(4) Assume that $s' > 1$. Use Euclid's algorithm to find integers σ and τ such that $\sigma s \equiv 1 \pmod{s'}$ and $\tau(-p) \equiv 1 \pmod{s'}$. This is possible because s and p are π-numbers, while s' is a π'-number. Let $m = (h^{-1}k)^{\sigma s}$ and $z = (m^h (m^2)^{h^2} \ldots (m^{p-1})^{h^{p-1}})^\tau$. Set $x = yz$ and stop.

The correctness of the conjugation algorithm is proved as follows. Without loss of generality assume that $y = 1$, and H and K share a common normal subgroup L, of prime index p. If $S = \langle H, K \rangle$, then L is normal in S, as it is normal in both H and K. Since $K = \langle k, L \rangle$, $S = \langle H, k \rangle$. However, $h^{-1}k = lm$ so that $S = \langle H, m \rangle = HM$, where $M = S \cap N_r$ is the normal closure of $\langle m \rangle$ in S. Since $L \cap M = \langle 1 \rangle$, it follows that $[L, M] = \langle 1 \rangle$. Now $h^{-1}k \in S \cap G_j = L \times M$, so if $s' = 1$, then $h^{-1}k \in L$ and so $H = K$. However, if $s' > 1$, then s' is the exponent of M, and $m = (hk^{-1})^{\sigma s}$ is the component of $h^{-1}k$ in M. Let $l = (h^{-1}k)m^{-1}$ be the component of $h^{-1}k$ in L.

Let ϕ be the linear transformation of the vector space M induced via conjugation by h. Then ϕ has order p because $h^p \in L$, and L centralises M. By Gorenstein (1968), Theorem 5.2.3, $M = C_M(H) \times [H, M]$ where both $C_M(H)$ and $[H, M]$ are normal in S. If $m = m_1 m_2$ where $m_1 \in C_M(H)$ and $m_2 \in [H, M]$, then $z = z_1 z_2$ where $z_1 \in C_M(H)$ and $z_2 \in [H, M]$. However, $H^z = H^{z_2}$ so there is no loss of generality in assuming that $C_M(H)$ is trivial.

Since $C_M(H)$ is trivial, $\sum_{i=0}^{p-1} \phi^i$ is the zero transformation of M. Using additive notation and writing linear transformations on the right gives

$$\begin{aligned} z(I-\phi) &= \left(\tau m \sum_{i=1}^{p-1} i\phi^i\right)(I-\phi) \\ &= \tau m\left(\sum_{i=1}^{p-1} i\phi^i - \sum_{i=2}^{p} (i-1)\phi^i\right) \\ &= \tau m\left(\sum_{i=1}^{p-1} \phi^i - (p-1)\phi^p\right) \\ &= \tau m(-I-(p-1)I) \\ &= \tau(-p)m \\ &= m. \end{aligned}$$

Therefore $[h, z] = m$ or $h^{-1}h^z = h^{-1}kl^{-1}$. Premultiplying by h gives $h^z = kl^{-1}$. Hence, $H^z \equiv K \pmod{L}$ and the correctness of the algorithm follows.

An alternate, though equivalent, method of finding z is to solve the linear equation $[h, z] = m$ for $z \in M$. This method was used in 3.2, whereas the explicit formula for z is due to Kantor (1985).

If H and K are arbitrary conjugate subgroups of G, then it may no longer be true that H and K are conjugate by an element of N_r. To generalise the conjugation algorithm, more knowledge of the cohomology group $H^1(H, N_r)$ is needed. The fact that this cohomology group is trivial when $|H|$ and $|N_r|$ are coprime accounts for the success of the previous algorithms.

If H and K are $\mathscr{F}$-projectors for some saturated formation $\mathscr{F}$, see Gaschutz (1963), then the conjugation algorithm can be used to find a conjugating element. In particular, H and K could be Carter subgroups of G.

Kantor & Taylor (1987) noted that a conjugation algorithm may be used to construct Hall π-subgroups.

THE HALL π-SUBGROUP ALGORITHM

Input: A finite soluble group G.
Output: A Hall π-subgroup H of G.

(1) If G is a π-group set $H = G$, and if G is a π'-group set $H = \langle 1 \rangle$, and stop.

(2) Recursively construct a Hall π-subgroup K/N_r of G/N_r. If N_r is a π-subgroup, then set $H = K$ and stop. (Henceforth, assume that N_r is a π'-subgroup and that $k_1, \ldots, k_s$ is an induced sequence for K.)

(3) Recursively construct a Hall π-subgroup M of $L = \langle k_2, \ldots, k_s \rangle$. If $p = |K:L| \notin \pi$, then set $H = M$ and stop.

(4) If $p \in \pi$, then use the conjugation algorithm to find $y \in N_r$ such that $(M^{k_1})^y = M$.

(5) Factorise the order of $k_1 y$ into a π-part ρ and a π'-part σ. Set $H = \langle (k_1 y)^\sigma, M \rangle$ and stop.

The correctness of the Hall π-subgroup algorithm follows from Lemma 3 and Theorem 4.

4. Constructing Normalisers of Hall π-subgroups

The conjugation and centraliser algorithms have applications to the construction of system normalisers, relative system normalisers, normalisers of Hall π-subgroups and Carter subgroups. These applications are presented after reviewing some terminology.

A Hall $\{p\}'$-subgroup of G is called a *Sylow p-complement* of G. Let $\{q_1, \ldots, q_s\}$ be the set of positive prime divisors of $|G|$. Then a set $\{H_1, \ldots, H_s\}$ is called a *complement basis* for G if each H_i is a Sylow q_i-complement. If $\{K_1, \ldots, K_s\}$ is another complement basis, then there exists a $g \in G$ such that $H_i^g = K_i$ for $1 \leqslant i \leqslant s$. An element g is found as follows. If G is a p-group, or more generally if G is nilpotent, then set $g = 1$. If G is not nilpotent, then recursively find $x \in G$ such that $H_i^x \equiv K_i \pmod{N_r}$ for $1 \leqslant i \leqslant s$. This is possible because $\{H_1 N_r/N_r, \ldots, H_s N_r/N_r\}$ is a complement basis for G/N_r. Let q_j be the exponent of N_r, then $N_r \leqslant H_i$ for $i \neq j$. Use the conjugation algorithm to find $y \in N_r$ such that $(H_j^x)^y = K_j$. Set $g = xy$, then $H_i^g = K_i$ for $1 \leqslant i \leqslant s$.

This simple algorithm may be generalised to construct system normalisers. The *system normaliser* of the complement basis $\Sigma = \{H_1, \ldots, H_s\}$ is defined to be

$$N(\Sigma) = \bigcap_{i=1}^{s} N_G(H_i).$$

THE SYSTEM NORMALISER ALGORITHM

```
Input:  A complement basis $\Sigma = \{H_1, \ldots, H_s\}$ for $G$.
Output: The system normaliser $N(\Sigma)$.
(1)     If $G$ is known to be nilpotent, for example, if $G$ is a $p$-group, then set $N(\Sigma) = G$
        and stop.
(2)     Recursively construct an induced sequence $x_1 N_r, \ldots, x_t N_r$ for the system
        normaliser
            $N(\Sigma N_r/N_r) = \bigcap_{i=1}^{s} N_{G/N_r}(H_i N_r/N_r)$.
(3)     Let $q_j$ be the exponent of $N_r$. Then for $1 \leqslant i \leqslant t$, use the conjugation algorithm to
        find $y_i \in N_r$ such that $(H_j^{x_i})^{y_i} = H_j$.
(4)     Use the centraliser algorithm to find an induced sequence $z_1, \ldots, z_u$ for $C_{N_r}(H_j)$.
        Then $x_1 y_1, \ldots, x_t y_t, z_1, \ldots, z_u$ is an induced sequence of $N(\Sigma)$.
```

The correctness of this algorithm is proved as follows. Since $N_r \leqslant H_i$ for $i \neq j$, it follows that $K = \langle x_1 y_1, \ldots, x_t y_t, z_1, \ldots, z_u \rangle$ is contained in $N(\Sigma)$. However,

$$N(\Sigma N_r/N_r) = N(\Sigma)N_r/N_r = KN_r/N_r \quad \text{and} \quad N(\Sigma) \cap N_r = K \cap N_r = C_{N_r}(H_j),$$

so using the second isomorphism theorem $N(\Sigma)/C_{N_r}(H_j) \cong K/C_{N_r}(H_j)$ and $K = N(\Sigma)$ as claimed.

The system normaliser algorithm is almost identical to an algorithm for constructing the normaliser of a Hall π-subgroup.

THE HALL π-NORMALISER ALGORITHM

```
Input:  A Hall $\pi$-subgroup $H$ of $G$.
Output: The normaliser $N_G(H)$.
(1)     Find the smallest integer $i$ such that $N_i$ is a $\pi$-subgroup of $G$. If $i \leqslant r$, then
        recursively construct $K/N_i = N_{G/N_i}(H/N_i)$. Set $N_G(H) = K$ and stop.
```

(2) Assume that N_r is a π'-subgroup of G. Recursively construct an induced sequence $x_1 N_r, \ldots, x_t N_r$ for $N_{G/N_r}(HN_r/N_r)$. Since H^{x_i} and H are Hall π-subgroups of HN_r, use the conjugation algorithm to find $y_i \in N_r$ such that $(H^{x_i})^{y_i} = H$.

(3) Use the centraliser algorithm to find an induced sequence $z_1, \ldots, z_u$ for $C_{N_r}(H)$. Then $x_1 y_1, \ldots, x_t y_t, z_1, \ldots, z_u$ is an induced sequence for $N_G(H)$.

The correctness of the Hall π-normaliser algorithm is proved in a similar manner to the correctness of the system normaliser algorithm. Both of these algorithms can be generalised to accommodate normal subgroups. For example, if H is a Hall π-subgroup of a normal subgroup N of G, and $x_1 N, \ldots, x_t N$ generate G/N, then a modification of the conjugation algorithm can be used to find $y_i \in N$ such that $(H^{x_i})^{y_i} = H$. The Hall π-normaliser algorithm can be used to find an induced sequence $z_1, \ldots, z_u$ for $N_N(H)$. Therefore $x_1 y_1, \ldots, x_t y_t, z_1, \ldots, z_u$ generates $N_G(H)$.

If $H_1, \ldots, H_s$ is a complement basis for N, then the *relative system normaliser* $\bigcap_{i=1}^{s} N_G(H_i)$, of N in G may be calculated similarly. The y_i being found by conjugating complement bases of N, as described above. It is particularly useful to be able to construct normalisers of the form $N_G(H)$, where H is a Hall π-subgroup of N, and relative system normalisers, because many complements to normal subgroups have these forms (Carter, 1961*b*).

The Hall π-normaliser algorithm may be used to construct Carter subgroups. In 1961, Carter proved the existence and conjugacy of nilpotent self-normalising subgroups of a finite soluble group. These subgroups, now called Carter subgroups, aroused considerable interest due partly to an analogy with the Cartan subalgebras of a Lie algebra. (Finite dimensional Lie algebras possess a single conjugacy class of nilpotent self-idealising subalgebras called Cartan subalgebras.) Subsequent work by Gaschütz (1963) on the theory of formations made precise the notion that a Carter subgroup is a "generalised" Hall π-subgroup. The ideas behind the Carter subgroup algorithm are embodied in Theorems 4, 5 and 6 (Carter, 1961*a*, Section 3).

THE CARTER SUBGROUP ALGORITHM

Input: A finite soluble group G.

Output: A Carter subgroup C of G.

(1) If G is a p-group, or more generally if G is nilpotent, then set $C = G$ and stop.

(2) Recursively construct a Carter subgroup K/N_r of G/N_r.

(3) If the exponent of N_r is q, then use the Hall π-subgroup algorithm to construct a Hall $\{q\}'$-subgroup H of K.

(4) Use the Hall π-normaliser algorithm to construct $N_K(H)$. Set $C = N_K(H)$ and stop.

5. Performance and Complexity

The author has implemented the Hall π-subgroup algorithm and the Hall π-normaliser algorithm in FORTRAN. These implementations will be included in the group theory program CAYLEY (Cannon, 1982) and are similar, but not identical, to the algorithms of this paper. Some run-time statistics, obtained on a VAX 11/780, are shown in Tables 1 and 2 below. The programs were run on two groups. The first group was the wreath product $((C_7 \, wr \, C_5) \, wr \, C_3) \, wr \, C_2$, of four cyclic groups and had order $2^1 3^2 5^6 7^{30}$. The second group was the wreath product $(S_4 \, wr \, S_4) \, wr \, S_4$, of three symmetric groups and had order $2^{63}3^{21}$. The *pc*-presentations used for these groups had very short relations. Had there been many long relations in the *pc*-presentations, the timings could have been up to ten times slower.

The symbols used in Tables 1 and 2 are explained below. Given a group G and a set π of primes, t_{Hall} and t_{norm} denote the times taken by the Hall π-subgroup algorithm to construct a Hall π-subgroup H, and the Hall π-normaliser algorithm to construct $N_G(H)$. The number of calls made to the conjugation algorithm by the Hall π-subgroup algorithm, and the total time spent by the conjugation algorithm are denoted by n_{conj} and t_{conj}, respectively. Similarly, n_{cent} and t_{cent} relate to the calls to the centraliser algorithm by the Hall π-normaliser algorithm.

Table 1. Running times for the Hall π-subgroup algorithm

$\lvert G\rvert$	π	$\lvert H\rvert$	t_{Hall} (sec)	n_{conj}	$t_{\text{conj}}/t_{\text{Hall}}$
$2^1 3^2 5^6 7^{30}$	$\{2\}$	2^1	0·03	0	0·00
$2^1 3^2 5^6 7^{30}$	$\{3\}$	3^2	0·74	2	0·45
$2^1 3^2 5^6 7^{30}$	$\{5\}$	5^6	1·82	5	0·48
$2^1 3^2 5^6 7^{30}$	$\{2, 5\}$	$2^1 5^6$	2·19	6	0·50
$2^1 3^2 5^6 7^{30}$	$\{3, 5\}$	$3^2 5^6$	2·59	7	0·50
$2^1 3^2 5^6 7^{30}$	$\{2, 3, 5\}$	$2^1 3^2 5^6$	2·94	8	0·50
$2^{63} 3^{21}$	$\{2\}$	2^{63}	62·57	36	0·50
$2^{63} 3^{21}$	$\{3\}$	3^{21}	46·92	28	0·47

Table 2. Running times for the Hall π-normaliser algorithm

$\lvert G\rvert$	π	$\lvert N_G(H)\rvert$	t_{norm} (sec)	n_{cent}	$t_{\text{cent}}/t_{\text{norm}}$
$2^1 3^2 5^6 7^{30}$	$\{2\}$	$2^1 3^1 5^3 7^{15}$	1·38	3	0·86
$2^1 3^2 5^6 7^{30}$	$\{3\}$	$2^1 3^2 5^2 7^{10}$	2·00	3	0·88
$2^1 3^2 5^6 7^{30}$	$\{5\}$	$2^1 3^2 5^6 7^6$	3·41	3	0·86
$2^1 3^2 5^6 7^{30}$	$\{2, 5\}$	$2^1 3^1 5^6 7^3$	3·31	2	0·95
$2^1 3^2 5^6 7^{30}$	$\{3, 5\}$	$2^1 3^2 5^6 7^2$	3·46	2	0·94
$2^1 3^3 5^6 7^{30}$	$\{2, 3, 5\}$	$2^1 3^2 5^6 7^1$	3·35	1	0·99
$2^{63} 3^{21}$	$\{2\}$	2^{63}	5·19	3	0·95
$2^{63} 3^{21}$	$\{3\}$	$2^7 3^{21}$	30·35	6	0·55

To gain insight into the complexity of the general normaliser problem, we compare it with the discrete logarithm problem (Coppersmith, 1984). Let $G = XYZ$ be a group of semilinear transformations of the field $GF(2^r)$, with 2^r elements. Let $X = \langle x \rangle$, where x is the squaring automorphism, $Y = \langle y \rangle$ where y generates the multiplicative group, and let Z be the additive group of $GF(2^r)$. The order of G is $r(2^r - 1)2^r$.

We construct an example for $r = 3$ using the irreducible polynomial $f(x) = x^3 + x + 1$. If w is a root of $f(x)$ in $GF(2^3)$, then $1, w, w^2$ generate Z. Identify y with w and z_0, z_1, z_2 with $1, w, w^2$. Then it follows that $y^x = y^2$, $z_0^x = z_0$, $z_1^x = z_2$ and $z_2^x = z_1 + z_2$, since $w^4 = w + w^2$. Similarly, $z_0^y = z_1$, $z_1^y = z_2$ and $z_2^y = z_0 + z_1$. Thus a *pc*-presentation for G is

$$\begin{aligned}\langle x, y, z_0, z_1, z_2 \mid x^3 &= y^7 = z_0^2 = z_1^2 = z_2^2 = 1,\\ &[y, x] = y,\ [z_0, x] = 1,\ [z_1, x] = z_1 z_2,\ [z_2, x] = z_1,\\ &[z_0, y] = z_0 z_1,\ [z_1, y] = z_1 z_2,\ [z_2, y] = z_0 z_1 z_2,\\ &[z_j, z_i] = 1,\ 0 \leqslant i < j \leqslant 2\rangle.\end{aligned}$$

Let h be a non-zero element of Z. Then calculating the normaliser $N_G(\langle h\rangle) = C_G(h)$ is equivalent to finding $C_{XY}(h)$ since $C_G(h) = C_{XY}(h)Z$. Since Y conjugates h to every other non-zero element of Z, $|XY : C_{XY}(h)| = 2^r - 1$. Thus $C_{XY}(h) = \langle xy^i\rangle$, for some $0 \leqslant i < 2^r - 1$. Suppose w generates the multiplicative group of $GF(2^r)$, then identify y with w, and h with the vector $h_0 + h_1 w + \ldots + h_{r-1} w^{r-1}$. If $*$ denotes field multiplication, then xy^i centralises h, when $h = h^{xy^i} = (h * h)^{y^i} = (h * h) * w^i$. Thus $h_0 + h_1 w + \ldots + h_{r-1} w^{r-1} = w^{-i}$ and calculating $N_G(\langle h\rangle)$ is equivalent to finding $-i$, the discrete logarithm of h.

The most efficient algorithm currently known for computing discrete logarithms in $GF(2^r)$ is due to Coppersmith (1984) and has asymptotic running time $O(\exp(cr^{1/3}\log^{2/3} r))$, where c is a constant. Indeed, since calculating discrete logarithms is difficult, many cryptographic schemes require the evaluation of discrete logarithms for code-breaking.

If $m = \log|G|$, then it can be shown that the number of group multiplications required by the sifting, centraliser, conjugation, Hall π-subgroup and Hall π-normaliser algorithms is $O(mn)$, $O(mn^2)$, $O(mn^3)$, $O(mn^4)$ and $O(mn^5)$, respectively. It seems unlikely that there exists a general purpose normaliser algorithm which is $O(mn^k)$ for some fixed k. If there were such an algorithm, then there would be an algorithm for finding discrete logarithms in $GF(2^r)$ which is polynomial in r. This follows since, for $G = XYZ$, $|G| = r(2^r - 1)2^r$ so $m < 3r$ and $n < 3r$, and $O(mn^k)$ is $O(r^{k+1})$.

I am indebted to Dr J. J. Cannon and Dr D. E. Taylor for their help and guidance, to Dr C. R. Leedham-Green for his inspiring conversations, to Mr J. H. Brownie for his assistance with programming, and to the referees for their helpful suggestions.

References

Cannon, J. J. (1982). (Preprint). *A Language for Group Theory*. University of Sydney, Australia.
Carter, R. W. (1961*a*). Nilpotent self-normalizing subgroups of soluble groups. *Maths. Z.* **75**, 136–139.
Carter, R. W. (1961*b*). Splitting properties of soluble groups. *J. London Math. Soc.* **36**, 89–94.
Coppersmith, D. (1984). Evaluating logarithms in $GF(2^n)$. In: *Proceedings 16th Annual Symposium on the Theory of Computing*. Pp. 201–207. New York: ACM.
Felsch, V. (1976). A machine independent implementation of a collection algorithm for the multiplication of group elements. In: *Proceedings of the 1976 ACM symposium on symbolic and algebraic computation*. Pp. 159–166. New York: ACM.
Gaschütz, W. (1963). Zur Theorie der endlichen auflösbaren Gruppen. *Math. Z.* **80**, 300–305.
Gorenstein, D. (1968). *Finite Groups*. New York: Harper and Row.
Hall, M., Jr. (1959). *The Theory of Groups*. New York: Macmillan Co.
Hall, P. (1969). *Nilpotent Groups*. (Notes of lectures given at the Canadian Mathematical Congress summer seminar, University of Alberta, 1957). London, Queen Mary College.
Jürgensen, H. (1970). Calculation with elements in a finite group given by generators and defining relations. *Computational Problems in Abstract Algebra*. Proc. Conf., Oxford, 1969. Oxford: Pergamon.
Kantor, W. M. (1985). Sylow's theorem in polynomial time. *J. Comp. Syst. Sci.* **30**, 359–394.
Kantor, W. M., Taylor, D. E. (1987). Polynomial-time versions of Sylow's theorem. *J. Algorithms* (in press).
Laue, R., Neubüser, J., Schoenwaelder, U. (1984). Algorithms for finite soluble groups and the SOGOS system. In: (Atkinson, M. D., ed.) *Computational Group Theory*. Proceedings of the London Mathematical Society Symposium on Computational Group Theory. Pp. 105–135. London: Academic Press.