CONSTRUCTING NORMALIZERS IN FINITE SOLUBLE GROUPS

S.P. Glasby

The University of Sydney, N.S.W., 2006, Australia

ABSTRACT

We describe an algorithm for constructing the normalizer, of certain subgroups, of a finite soluble group. Let G be a finite soluble group divisible by n primes, counting multiplicities. Then the normalizer algorithm terminates after $O(log|G|n^5)$ group mulitplications. This algorithm can construct the normalizer of a Hall π-subgroup and hence may be used to construct Carter subgroups.

The existing algorithms for constructing the normalizer of an arbitrary soluble group, have running times which are exponential in n. This is despite the fact that the normalizer of certain subgroups can be found in polynomial time. By considering a family of examples, we show that the problem of constructing normalizers for soluble groups is at least as difficult as the discrete logarithm problem for finite fields. There no known polynomial solution to the discrete logarithm problem.

§1 INTRODUCTION

There are three common ways of representing a finite soluble group: (1) as a permutation group acting on a set, (2) as a group of matrices over some ring, or (3) as a finitely presented group where the given relations constitute a power-commutator presentation (PC–presentation). The first two ways have the drawback that there is no computationally efficient method of representing an arbitrary quotient of a permutation or matrix group. There is, however, an easy method for constructing a PC–presentation for an arbitrary quotient of a group endowed with a PC–presentation. For this reason the normalizer algorithm is designed for groups endowed with PC–presentations.

A PC–presentation is a presentation of the form

$$G = \langle\, g_1, \ldots, g_n \,|\, g_i^{p_i} = u_{ii}, 1 \le i \le n, [g_j, g_i] = u_{ij}, 1 \le i < j \le n \,\rangle, \qquad (*)$$

where $u_{ij}, 1 \le i \le j \le n$, is a word of the form $g_{i+1}^{k_{i,j,i+1}} \ldots g_n^{k_{i,j,n}}$ with $0 \le k_{i,j,\ell} < p_\ell$ for $i < \ell \le n$. Our PC–presentations will additionally have each p_i prime and $|G| = p_1 \ldots p_n$.

Since any finite soluble group G has a subnormal series where the quotients of consecutive terms are cyclic of prime order, G has a PC–presentation. Conversely, if G is defined by the PC–presentation $(*)$ then the subgroups $G_i = \langle g_i, \ldots, g_n \rangle, 1 \le i \le n+1$, define a subnormal series with cyclic quotients. Therefore, G is finite and soluble.

It is easy to show that every element of G has a unique representation as a *collected* word $g_1^{k_1} \ldots g_n^{k_n}$, with $0 \le k_i < p_i$. The product of two collected words is expressed as a collected word using commutator collection, (Hall (1969), p.29). A particularly efficient algorithm for commutator collection is described by Jürgensen (1970).

An explicit example of a PC–presentation appears in section 4. In section 2 we give the reader some examples of basic algorithms for PC–presentation groups. Throughout this paper, all groups will be finite, soluble and endowed with PC–presentations.

§2 PRELIMINARY ALGORITHMS

The most interesting of the five algorithms below are A3 and A4. Algorithm A1 requires the concept of weight. The *weight* of $g \in G$, denoted $w(g)$, is defined to be the smallest i satisfying $g \in G_i$. Note that only the identity element has weight $n+1$.

A1 The sifting algorithm. The following algorithm generalizes the well-known Gaussian elimination algorithm, see Laue *et al.* (1984), p. 108.
Input: A sequence $h_1, \ldots, h_m$ of elements of G.
Output: An integer ℓ and a sequence $k_1, \ldots, k_m$ such that $\langle h_1, \ldots, h_m \rangle = \langle k_1, \ldots, k_\ell \rangle$
where $w(k_1) < \ldots < w(k_\ell) < w(k_{\ell+1}) = \ldots = w(k_m) = n+1$.

begin
$\ell := 0$;
for $i = 1$ to m do
$k_i := h_i$;
while $w(k_i) = w(k_j)$ for some $1 \leq j \leq \ell$ do
{ We now *sift* k_i through $k_j, \ldots, k_\ell$. }
let k_i and k_j be congruent modulo G_{s+1} to g_s^a and g_s^b respectively;
find c, prime to the order of h, such that $ac \equiv b (\mathrm{mod}\, p_s)$;
$k_i := k_i^{-c} k_j$; { Note that $w(k_i) < w(k_i^{-c} k_j)$.}
end while;
if $w(k_i) = n + 1$ { that is $k_i = 1$ } then
loop; { Choose the next value for i.}
end if;
$\ell := \ell + 1$;
let j be the largest integer less than i such that $w(k_j) \leq w(k_i)$;
we obtain $k_1, \ldots, k_i$ from $k_1, \ldots, k_{i-1}$ by inserting k_i after k_j;
end for;
end;

A sequence $h_1, \ldots, h_m$ of generators for the subgroup H of G is called a *sequence of PC–generators* for H if the subgroups $\langle h_i, \ldots, h_m \rangle$ define a maximal subnormal chain for H. If additionally, $w(h_1) < \ldots < w(h_m) < n + 1$ obtains, the sequence is called *induced*. We shall represent H by an induced sequence of PC–generators. For then if $g \in G$, we may sift g through $h_1, \ldots, h_m$ to determine whether or not $g \in H$. If $g \in H$, we may *rewrite* g as a collected word in the h_i.

A sequence $h_1, \ldots, h_m$ is an induced sequence of PC–generators for $H = \langle h_1, \ldots, h_m \rangle$ precisely when $w(h_1) < \ldots < w(h_m)$ and each h_i^p, $[h_j, h_i]$ may be rewritten as a collected word in $h_1, \ldots, h_m$. An algorithm, called the non-commutative Gauss algorithm (NCGA) (Laue *et al.* (1984), p.108), uses this idea, together with the sifting algorithm, to produce an induced sequence of PC–generators for H from an arbitrary set of generators for H.

A2 Constructing a π-separable normal series

The reader is referred to Bray *et al.* (1982), p.206 for definitions of π-numbers, π-groups and Hall π-subgroups. A normal series $G = N_0 \trianglerighteq \ldots \trianglerighteq N_{2d} = 1$ is called π-*separable* if N_{2i}/N_{2i+1} is a π'-group and N_{2i+1}/N_{2i+2} a π-group for each i.

If $g_1, \ldots, g_n$ are PC–generators for G then the derived group G' of G is generated by the set $\{ [g_j, g_i] | 1 \leq i < j \leq n \}$. Indeed, applying the sifting algorithm to this set produces an induced sequence of PC–generators for G'. In this manner we may construct the derived series.

We construct a π-separable normal series as follows.
begin
let $N_{2i} = G^{(i)}$ be the ith term in the derived series;
suppose $|N_{2i} : N_{2i+2}| = kk'$ where k is a π-number and k' a π'-number;
if $N_{2i} = \langle x_1, \ldots, x_m, N_{2i+2} \rangle$ then define N_{2i+1} to be $\langle x_1^k, \ldots, x_m^k, N_{2i+2} \rangle$;
end;

This algorithm and the NCGA can be used to create a PC–presentation for G with the following property : the subnormal series $G = G_1 \triangleright \ldots \triangleright G_{n+1} = 1$ refines a normal series $G = N_1 >$

$\ldots \rhd N_{r+1} = 1$ where N_i/N_{i+1} is an abelian q_i-group. Henceforth we shall assume our PC–presentations have this additional property.

A3 The centralizer algorithm.

Let H be a subgroup of G which normalizes the abelian subgroup N of G. When H is cyclic the following algorithm finds $C_N(H)$ and $[H, N]$, in a manner akin to finding the kernel and image of a linear map. This algorithm generalizes to find $C_N(H)$ for non-cyclic subgroups H of G.

Input: $H = \langle h \rangle$ and a sequence $y_1, \ldots, y_m$ of induced PC–generators for N.
Output: A sequence $y'_1, \ldots, y'_m$ of generators for N and an integer ℓ such that
$C_N(H) = \langle y'_{\ell+1}, \ldots, y'_m \rangle$ and $[h, y'_1], \ldots, [h, y'_\ell]$ is an induced sequence of PC–generators for $[H, N]$.

begin
rewrite $[h, y_i]$, $1 \leq i \leq m$, as a collected word x_i in $y_1, \ldots, y_m$;
use the sifting algorithm to transform $x_1, \ldots, x_m$ to $x'_1, \ldots, x'_m$. Use the same operations to transform $y_1, \ldots, y_m$ to $y'_1, \ldots, y'_m$;
{Note that the map taking y to $[h, y]$ defines a linear endomorphism of N.
Thus $[h, y_i] = x_i$ implies $[h, y'_i] = x'_i$.}
let ℓ be the largest integer satisfying $w(x'_\ell) < n + 1$;
end;

If H is an arbitrary subgroup of G with PC–generators $h_1, \ldots, h_m$, we may calculate $C_N(H)$ as follows. Recursively calculate $M = C_N(\langle h_2, \ldots, h_m \rangle)$. Then, because M is abelian and normalized by h_1, we may find $C_M(\langle h_1 \rangle) = C_N(H)$.

A4 The conjugation algorithm

Let H be a subgroup and Q an abelian subgroup of G. Suppose H normalizes Q and $H \cap Q = 1$. If K is some conjugate in Q of H, we find $y \in Q$ such that $H^y = K$. Kantor and Taylor have described a different algorithm for conjugating Sylow subgroups of permutation groups.

Input: H, K, Q as above and a subnormal series $HQ = G_1 \rhd \ldots \rhd G_s = Q \rhd \ldots \rhd G_{n+1} = 1$.
Output: An element y of Q with $H^y = K$.

begin
$y := 1$;
while $H \neq K$ do
{ Denote the subgroups $G_j \cap H$ and $G_j \cap K$ by H_j and K_j respectively.}
construct H_j and K_j using a variant of A1;
let i be the largest integer such that $H_i \neq K_i$;
use algorithm **conj** to find $y_i \in Q$ such that $H_i^{y_i} = K_i$;
$H := H^{y_i}; y := y y_i$;
end while;
end;

Algorithm conj

Input: Conjugate subgroups H_i and K_i of H_iQ with $H_{i+1} = K_{i+1}$.
Output: An element $y_i \in Q$ satisfying $H_i^{y_i} = K_i$.

begin
    find $h \in H$, $k \in K$ such that h, k are congruent to g_i modulo G_{i+1};
    {Since $h^{-1}k \in H_{i+1}Q = K_{i+1}Q, h^{-1}k = \ell m$ with $\ell \in K_{i+1}$ and $m \in Q$. }
    use a variant of A1 to express $h^{-1}k$ as ℓm with $\ell \in K_{i+1}$ and $m \in Q$;
    use A2 to find $y_i \in Q$ such that $[h, y_i] = m$;
end;

The above algorithm assumes that $[h, y_i] = m$ has a solution for $y_i \in Q$, and also that $H_i^{y_i} = K_i$. We prove these assumptions as follows. The subgroup K_{i+1} is normal in H_i and K_i, and hence normal in $S = \langle H_i, K_i \rangle$. Therefore $[K_{i+1}, S \cap Q] \leq K_{i+1} \cap (S \cap Q) \leq H \cap Q = 1$ and so $lm = ml$. Since H and K are conjugate in Q, H_i and K_i are conjugate by an element, y_i say, of Q. Since $h^{-1}k = \ell m$, $h = km^{-1}\ell^{-1}$ so $h^{y_i} \equiv h \equiv k\ell^{-1} \bmod Q$. However h^{y_i} and $k\ell^{-1}$ are elements of K_i so $h^{y_i} = k\ell^{-1}$. Premultiplying this equation by h^{-1} gives $[h, y_i] = h^{-1}k\ell^{-1}$. But $h^{-1}k = \ell m = m\ell$ so $h^{-1}k\ell^{-1} = m$. Thus, the equation $[h, y_i] = m$ has a solution. Furthermore, as any solution y_i satisfies $h^{y_i} = k\ell^{-1}$, we have $H_i^{y_i} = K_i$.

A5 Constructing Hall π-subgroups As Kantor and Taylor noted, a conjugation algorithm may be used to construct Hall π-subgroups.
Input: A group G and a set π of primes.
Output: A Hall π-subgroup H of G.

begin
recursively construct a Hall π-subgroup H_2 of G_2;
if $p = |G : G_2| \notin \pi$ set $H = H_2$;
{ Henceforth assume that $p \in \pi$. }
use the conjugation algorithm (A4) to find $y \in G_2$ such that $(H_2^{g_1})^y = H_2$;
let h_1 be the p-part of $g_1 y$ then set $H = \langle h_1, H_2 \rangle$;
end;

§3 CARTER SUBGROUPS AND NORMALIZERS OF HALL π-SUBGROUPS

The conjugation algorithm may be used to compute the normalizer of any subgroup H with the property that for $1 \leq i \leq r$ either $N_i \leq HN_{i+1}$ or $H \cap N_i \leq N_{i+1}$ and $N_G(HN_i/N_i) = N_G(HN_{i+1}/N_{i+1})N_i$. This property holds if H Hall π-subgroup of some N_j, or if H is an $\mathcal{F}$-projector for some saturated formation $\mathcal{F}$ (Bray *et al.* (1982), p.143). To be concrete we shall describe the normalizer algorithm when H is a Hall π-subgroup of G. A paper by Alperin (1964) motivates the following question. Do system normalizers satisfy the above property for an appropriate choice of N_i ? If they do, then the normalizer algorithm would be a useful research tool.

Input: A set π of primes, a soluble group G and a Hall π-subgroup H of G constructed using A5 or otherwise.
Output: The normalizer $N_G(H)$.

begin
let $Q = N_r$;
recursively calculate $N_G(HQ/Q)$, the normalizer of H modulo Q;
if Q is a π-subgroup then $N_G(H) = N_G(HQ/Q)$ so stop;
{Assume now that Q is an abelian π'-subgroup. }
let $k_1, \ldots, k_s$ generate $N_G(HQ/Q)$ modulo HQ;
use the conjugation algorithm to find $y_i \in Q$ such that $(H^{k_i})^{y_i} = H$;
use A3 to find $C_Q(H)$;
$K = \langle k_1 y_1, \ldots, k_s y_s, H \times C_Q(H) \rangle$ equals $N_G(H)$ so stop;
end;

Note that K clearly normalizes H. Furthermore, if $g \in N_G(H)$ is congruent to $k_1^{\ell_1} \ldots k_s^{\ell_s}$ modulo HQ, then $g^{-1}(k_1 y_1)^{\ell_1} \ldots (k_s y_s)^{\ell_s} \in N_{HQ}(H) = H \times C_Q(H)$. Thus, $g \in K$ so $K = N_G(H)$ as claimed.

The above algorithm may be used to compute Carter subgroups, (ie. nilpotent self-normalizing subgroups). While an arbitrary group may not possess a Carter subgroup, every finite soluble group does. The algorithm presented below will construct a Carter subgroup of a finite soluble group. (Note that Carter subgroups form a single conjugacy class, (Carter (1961), Section 4).)

We construct a Carter subgroup of G as follows. Recursively construct a Carter subgroup K/Q of G/Q, where $Q = N_r$ is an abelian normal q_r-subgroup of G. Use A5 to find a Hall q_r'-subgroup H of K. Then $N_K(H)$ is a Carter subgroup of G. (A proof of this fact follows from Carter (1961), Section 3, with a slight modification.)

§4 COMPLEXITY

The normalizer of an arbitrary subgroup H of G may be calculated by an *orbit–stabilizer* calculation. (G acts via conjugation on the set of conjugates of H , the stabilizer of H is $N_G(H)$.) Laue *et al.* (1984) p.111 use this idea to calculate normalizers in soluble groups. This algorithm can require an exponential amount of time and space to find $N_G(H)$. By contrast the algorithm of the previous section has polynomial time and space complexity when applied to the subgroups satisfying the conditions of the previous section.

The author has implemented most of the above algorithms in FORTRAN and their performance seems encouraging. For example, if G is the group S_4 wr S_4 wr S_4 of order $2^{63}3^{21}$, we may find a Hall $\{3\}$-subgroup in 47 seconds and its normalizer, which has order $2^7 3^{21}$, in 32 seconds. (The timings were obtained using a VAX 11/780 computer.)

To gain insight into the complexity of the general normalizer problem we compare it with the discrete logarithm problem, (see 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

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

Let h be a nonzero element of Z. The calculation of the normalizer $N_G(\langle h \rangle) = C_G(h)$ is equivalent to finding $C_{XY}(h)$ since $C_G(h) = C_{XY}(h)Z$. Since Y conjugates h to every other nonzero element of Z, $|XY : C_{XY}(h)| = 2^r - 1$. Thus, $C_{XY}(h) = \langle xy^i \rangle$, for some i, $0 \le i < 2^r - 1$. 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 centralizes 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 the calculation of $N_G(\langle h \rangle)$ 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 small

constant. Indeed, since the calculation of discrete logarithms is difficult, many cryptographic schemes require the evaluation of discrete logarithms for code-breaking.

If $m = log|G|$ then the sifting algorithm requires $O(mn)$ group element multiplications. In fact, A2, ..., A5 and the algorithm for finding normalizers of Hall π-subgroups are all $O(mn^k)$ where $k \leq 5$. It seems unlikely that there exists a general normalizer algorithm which is $O(mn^k)$ for some k. For then there would be an algorithm, polynomial in r, for finding discrete logarithms in $GF(2^r)$. Note that for $G = XYZ, |G| = r(2^r - 1)2^r$ so $m < 3r$ and $n < 3r$, and $O(mn^k)$ is $O(r^{k+1})$.

Acknowledgements I am in debted 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, and to Mr. J. Brownie for his assistance with programming.

REFERENCES

Alperin, J.L.(1964).Normalizers of system normalizers. *Trans. Amer. Math. Soc.*, 181-188.

Bray, H.G. *et al.* (1982). "Between Nilpotent and Solvable," (Weinstein M., Ed.), Polygonal Publishing House, New Jersey, U.S.A. .

Cannon, J.J. (1982) (Preprint). A Language for Group Theory , University of Sydney, Australia.

Carter, R.W. (1961). Nilpotent self-normalizing subgroups of soluble groups . *Math. Z.* 75, 136-139.

Coppersmith, D. (1984). Evaluating logarithms in $GF(2^n)$, *in* "Proceedings 16th Annual Symposium on the Theory of Computing," 201-207, ACM, New York.

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 of a finite group given by generators and defining relations, *in* "Computational Problems in Algebra" (Leech J., Ed.), 47-57, Pergamon Press, Oxford.

Kantor, W.M. and Taylor, D.E. .Polynomial-time versions of Sylow's theorem. *J. Algorithms* (to appear).

Laue, R., Neubüser, J. and Schoenwaelder, U. (1984). Algorithms for finite soluble groups and the SOGOS system, *in* "Computational Group Theory" (Atkinson, M.D., Ed.), 105-135, Proceedings of the London Mathematical Society Symposium on Computational Group Theory, Academic Press.