Efficient multiplication algorithms for finite polycyclic groups

by Burkhard Höfling

Technische Universität, Institut für Geometrie,
Pockelsstr. 14, 38106 Braunschweig, Germany
e-mail: `b.hoefling@tu-bs.de`

Abstract: Let G be a finite soluble group given by a reduced confluent polycyclic presentation. Represent group elements by reduced words. Then there exists an algorithm for multiplying two group elements which has subexponential running time and requires polynomial space. Moreover, given the prime factorisation of the group order, the problem of multiplying two group elements is probabilistically polynomial time equivalent to the same problem for p-groups, where p divides the order of G.

Classification (AMS 2000): 20-04, 20D10, 68W40

1. Introduction

A *polycyclic presentation* $\mathscr{P}$ of a (necessarily soluble) finite group G is a presentation of the form

$$\langle\, g_1, \ldots, g_n \mid g_i^{e_i} = w_{i,i}, 1 \le i \le n, g_j g_i = g_i w_{i,j}, 1 \le i < j \le n \,\rangle,$$

where, for $1 \le i \le j \le n$, the e_i are integers ≥ 2 and the $w_{i,j}$ are words in $g_{i+1}, \ldots, g_n$. Regarding the above relations as rewriting rules $g_i^{e_i} \to w_{i,i}$, $g_j g_i \to g_i w_{i,j}$, we obtain a Knuth Bendix rewriting system for G with respect to the wreath product ordering; see [11, Section 9.4]. The rewriting process is usually called *collection*. The resulting reduced expression for $g \in G$ with respect to this system is of the form $g = g_1^{a_1} \ldots g_n^{a_n}$, where $0 \le a_i < e_i$. We call $\mathscr{P}$ *reduced* if the $w_{i,j}$ are reduced; it is *consistent* if the corresponding Knuth Bendix rewriting system is confluent, that is, if the a_i are uniquely defined by g. This is clearly the case if and only if $|G| = e_1 \cdots e_n$. In this case, we will refer to $g_1^{a_1} \ldots g_n^{a_n}$ as the *normal form* or *reduced form* of g. Throughout the article, we will assume that all polycyclic presentations are reduced and consistent.

the process of finding the reduced form of the product $w_1 w_2$, where w_1 and w_2 are reduced words in $g_1, \ldots, g_n$, will be called a *multiplication* with respect to $g_1, \ldots, g_n$ (or with respect to $\mathscr{P}$). A *(Las Vegas) multiplication algorithm* with respect to $g_1, \ldots, g_n$ is a (Las Vegas) algorithm which carries out such a multiplication. Note that every multiplication algorithm can compute a reduced expression for the inverse of an element, see Lemma 3.3. Using this fact, a multiplication algorithm can be used to reduce any word w in $g_1, \dots, g_n$ to its normal form by replacing inverses and multiplying reduced subwords.

The possibly simplest multiplication algorithm consists in collecting the concatenation of the words to be multiplied. When collecting a word, usually several rewriting rules are applicable at each step. The efficiency of collection depends heavily on the strategy used to make a choice, and it is now generally believed that collection from the left is the best such strategy, at least for practical purposes; see [9, 12]. For nilpotent groups and p-groups, more elaborate multiplication algorithms are known; see [10]; note that some authors use the term "collection" in the more general sense of "multiplication".

While algorithms such as collection from the left work very well in many practical situations, [9] contains easy examples of presentations of cyclic groups of prime power order for which the number of collection steps in any collection algorithm grows exponentially in the composition length of the group. On the other hand, such a group has an obvious presentation such that the number of collection steps only grows linearly in the composition length. This indicates that the performance of a collection algorithm not only depends on the strategy but also on the presentation of a given group G.

Therefore, in the present article, we use the following approach for finding efficient multiplication algorithms. In order to multiply two words which are reduced with respect to $\mathscr{P}$, compute a presentation $\mathscr{P}'$ of G for which a fast multiplication algorithm is known. Then translate the words to be multiplied into reduced words with respect to $\mathscr{P}'$. Multiply then, using the multiplication algorithm for $\mathscr{P}'$, and translate the result back into a reduced word with respect to $\mathscr{P}$. Such an approach requires a preprocessing step for computing $\mathscr{P}'$ and possibly additional information needed for subsequent translations of reduced words. This will be visible in the results below.

In Sections 3 – 5, we let $\mathscr{P}'$ reflect the structure of the abelian factors in the derived series of G and use a slight variant of collection from the left and obtain Theorem 5.1, which can be summarised as follows. Details about our computational model can be found in Section 2.

Theorem. *Let G be a finite soluble group of derived length d and composition length l, and let $\mathscr{P}$ be a reduced confluent polycyclic presentation of a finite group G. Then there exists a multiplication algorithm such that r multiplications require at most*

$$O((l^3 \log^2 |G| + r)(Cl \log |G|)^{2d} + dlL^6) \subseteq e^{C' \log l \log \log |G|}$$

bit operations, where C, C' are suitable constants, and $O(dl^2 \log |G|)$ bits of workspace.

Since the input includes $\mathscr{P}$ and therefore has a bit length of at least $\log_2 |G|$, this has the following consequence.

Corollary. *For any polycyclic presentation, there exists a multiplication algorithm which has subexponential running time and requires polynomial space.*

In Sections 6 – 9, we follow a different approach. We show that the problem of finding an asymptotically fast multiplication algorithm for an arbitrary soluble group reduces to finding such an algorithm for p-groups, provided the prime factorisation of the e_i is known. This reduction is carried out in four steps. Proposition 6.1 shows that we may assume the e_i to be primes, and by Theorem 6.2, it suffices to consider presentations where a subset of the G_i forms a normal series with elementary abelian factors. Then we reduce to multiplications in nilpotent subgroups in Theorem 8.5, and finally from nilpotent subgroups to their Sylow p-subgroups in Theorem 9.1. Putting these results together, we obtain the following result; cf. Corollary 9.3.

Theorem. *Let $\mathscr{P}$ be a reduced confluent polycyclic presentation, and assume that the prime factorisation of each e_i is known. Then there exists a Las Vegas multiplication algorithm for $\mathscr{P}$ which requires an expected number of*

$$O(l^4\mathbf{P}_p(G) + l^9 \log^5 |G|\mathbf{M}_p(G) + rl^3 \log^3 |G|\mathbf{M}_p(G))$$

bit operations to perform r multiplications, where l is the composition length of G.

Here, $\mathbf{P}_p$ and $\mathbf{M}_p$ are functions modelling the cost of r multiplications in the Sylow p-subgroup of G, one for each prime, where $\mathbf{P}_p(G)$ is the cost of a preprocessing step and $\mathbf{M}_p(G)$ is the cost of a subsequent multiplication. A more precise definition of $\mathbf{P}_p$ and $\mathbf{M}_p$ can be found in Section 9.

Thus, the question whether there exists a polynomial time multiplication algorithm for $\mathscr{P}$ is reduced to the case of p-groups and the problem of factorising integers.

Corollary. *If there exists a (Las Vegas) multiplication algorithm for p-groups which runs in polynomial time, then there exists a Las Vegas polynomial time multiplication algorithm for any finite soluble group, given prime factorisations of the e_i.*

Note that it is an open problem whether integers can be factorised in polynomial time. Finding a factorisation of $e \in \mathbb{N}$ is obviously a prerequisite for reducing to p-groups, for it is equivalent to finding the Sylow subgroups of the cyclic group $\langle\, g \mid g^e = 1 \,\rangle$.

The only probabilistic part of the algorithm is the use of the meat-axe algorithm for finding submodules of a reducible matrix group [7, 8].

The bounds obtained above, especially the one in the second theorem, may seem to be too large for practical computations. However, it seems unlikely that in concrete examples, the worst case assumptions leading to them are justified. Indeed, since a multiplication in G is replaced by a large number of multiplications in (usually small) p-subgroups, this may still lead to a reduction of the overall multiplication time. We leave the question of practical usefulness for further investigations.

2. Notation and definitions

Throughout the article, G will be a finite soluble group with reduced confluent polycyclic presentation

$$\langle\, g_1, \dots, g_n \mid g_i^{e_i} = w_{i,i}, 1 \le i \le n, g_j g_i = g_i w_{i,j}, 1 \le i < j \le n \,\rangle.$$

The terms “multiplication”, “reduced form”, “normal word” etc. all refer to $\mathscr{P}$ (or its generators $g_1, \dots, g_n$, unless specified otherwise. All polycyclic presentations will be assumed reduced and confluent.

Our computational model is based on the following assumptions. The number of bit operations for adding, subtracting and multiplying two nonnegative integers $\le n$ is $O(\log n)$, $O(\log n)$ and $O(\log^2 n)$, respectively. Division with remainder and the Extended Euclidean algorithm require $O(\log^2 n)$ and $O(\log^3 n)$ bit operations, respectively. Storing a nonnegative integer $< n$ requires $O(\log n)$ bits, and copying it requires $O(\log n)$ bit operations.

We will also assume that a group multiplication in a group H requires at least $O(\log |H|)$ bit operations. This agrees with our assumptions on integer arithmetic and allows to ignore the cost of index arithmetic.

Let $l_i = \lfloor \log_2(e_i - 1) + 1 \rfloor$ and put $L_k = \sum_{i=k}^n l_i$ and $I_k = \sum_{i=k}^n l_i^3$. We set $L = L_1$ and $I = I_1$. Note that L is a lower bound on the input length of a multiplication algorithm for G, since at least L bits are required to store the exponents $e_1, \dots, e_n$ Moreover, a word in $g_1, \dots, g_n$ requires $O(L)$ bits of storage, so that $\mathscr{P}$ can be stored using $O(n^2 L) \subseteq O(L^3)$ bits of storage. Clearly, $L \in O(\log |G|)$ and $\log |G| \in O(L)$.

3. Variants of collection from the left

If $w = g_{i_1} g_{i_2} \cdots g_{i_r}$ is a word in $g_1, \dots, g_n$ which is not reduced, then *collection from the left* always chooses j minimal such that $g_{i_j} \cdots g_{i_k}$ is a left hand side of one of the relation in $\mathscr{P}$ for an integer $k \ge j$, and replaces $g_{i_j} \cdots g_{i_k}$ by the corresponding right hand side.

Easy examples show that the number of bit operations required to reduce a word $g_j^b g_i^a$ using collection to the left grows at least linearly in a and b. This leads to performance problems when e_i and e_j are large. It is well-known that the case when b is large can be handled by fast powering, and that a similar idea, namely computing powers of the conjugation automorphism induced by g_i, can be used if a is large; cf. e. g. [5]. The following algorithm uses a similar approach, and it is not difficult to see that all bounds in this section hold with a suitable modification of collection from the left.

3.1 Algorithm.
Input: integers i, k such that $1 \le i \le k \le n$ and such that g_i normalises G_{k+1}, reduced words $w_1, \dots, w_r \in G_{k+1}$, a nonnegative integer $b = \sum_{j=0}^{m-1} b_j 2^j$ with $b_i \in \{0, 1\}$.

Output: the normal forms v_l of $w_l^{g_i^{a-1}} w_l^{g_i^{a-2}} \cdots w_l$, where $1 \leq l \leq r$.

```
(1) let $x_{k+1} := w_{i,k+1}, \dots, x_n := w_{i,n}$;
(2) for $l = 1, \dots, r$ do let $u_l := w_l$, $v_l := 1$;
(3) for $j = 0, \dots, m-1$ do
(4)     if $b_j = 1$, then
(5)         for $l = 1, \dots, r$, write $v_l = g_{k+1}^{a_{k+1}} \cdots g_n^{a_n}$ and replace $v_l$ by the normal form
              of $x_{k+1}^{a_{k+1}} \cdots x_n^{a_n} u_l$.
(6)     for $l = 1, \dots, r$ do write $u_l = g_{k+1}^{a_{k+1}} \cdots g_n^{a_n}$ and let $u_l$ be the normal form of
          $x_{k+1}^{a_{k+1}} \cdots x_n^{a_n} u_l$;
(7)     for $l = k+1, \dots, n$ do write $x_j = g_{k+1}^{a_{k+1}} \cdots g_n^{a_n}$ and let $y_j$ be a reduced
          expression for $x_{k+1}^{a_{k+1}} \cdots x_n^{a_n}$.
(8)     for $l = k+1, \dots, n$ do let $x_l := y_l$;
```

3.2 Lemma. *Let i, k be integers with $1 \leq i \leq k < n$, and assume that g_i normalises G_{k+1}. Moreover, let $b \in \mathbb{N} \setminus \{0\}$ and $m = \lfloor \log_2 b \rfloor + 1$, and assume that $w_1, \dots, w_r \in G_{k+1}$ are reduced. Then the normal forms of*

$$w_l^{g_i^{b-1}} w_l^{g_i^{b-2}} \cdots w_l^{g_i} w_l, \qquad w_l w_l^{g_i} \dots w_l^{g_i^{b-1}}, \qquad w_l^{g_i^b},$$

where $1 \leq l \leq r$, can be computed using $O(m(n-k+r)L_{k+1})$ multiplications in G_{k+1} and $O((n-k+r)L_{k+1})$ bits of workspace.

Proof. We only prove the statement for

$$w_l^{g_i^{b-1}} w_l^{g_i^{b-2}} \cdots w_l,$$

for which Algorithm 3.1 is an algorithm, the other statements being similar. To prove the correctness of Algorithm 3.1, it suffices to consider the case when $r = 1$. Write $g = g_i$, $u = u_1$, $v = v_1$, $w = w_1$. For $t \in \mathbb{N}$, let $z_t := w^{g^{l-1}} w^{g^{l-2}} \cdots w$, and observe that $z_{s+t} = (z_s)^{g^t} z_t$ for all $s, t \in \mathbb{N}$. Now an easy induction argument shows that at the beginning of Step (4) of Algorithm 3.1, we have $x_l = g_l^{g^{2^j}}$, $u = z_{2^j}$, $v = z_{s_j}$, where $s_j = \sum_{l=0}^{l-1} b_j 2^j$, so that

$$(g_{k+1}^{a_{k+1}} \cdots g_n^{a_n})^{g^{2^j}} = x_{k+1}^{a_{k+1}} \cdots x_n^{a_n}. \tag{$*$}$$

Evaluating ($*$) using fast powering requires at most $2 \sum_{l=k+1}^{n} \lfloor 1 + \log_2 a_l \rfloor \in O(L_{k+1})$ multiplications in G_{k+1}. Storing a reduced word requires $O(L_1)$ bits. The statement now follows. □

In the case of collection to the left, it is well-known that the normal form w of $g^{-1}h$ can be computed at the same cost as the product gw; see also Proposition 3.4 (b). For other multiplication algorithms, this need not be the case, see, e. g. [10, Section 7.1] However, we obtain the following.

3.3 Lemma. *Let g, h be normal words in G. Then the normal form of $g^{-1}h$ can be computed using at most n multiplications in G and $O(L)$ bits of workspace.*

Proof. Let $g = g_1^{a_1} \dots g_n^{a_n}$, $h = g_1^{b_1} \dots g_n^{b_n}$ be the normal forms of g and h. Let $0 \le c < e_1$ be such that $c \equiv b_1 - a_1 \pmod{e_1}$. Compute the normal form $g_1^{d_1} \dots g_n^{d_n}$ of gg_1^c, using one multiplication in G. Let $g^* = g_2^{d_2} \dots g_n^{d_n}$, $h^* = g_2^{b_1} \dots g_n^{b_n}$. In G_2, recursively compute a normal form w^* such that $(g^*)w^* = h^*$, at a cost of $O(L_2)$ multiplications in G. Then $w = g_1^c w^*$ is in normal form, and one easily checks that $gw = h$, observing that $d_1 = b_1$. □

The complexity of the modified version of collection from the left can now be analysed as follows; note that the hypotheses are always satisfied for $k = 1$.

3.4 Proposition. *Let $1 \le k \le n$ be such that $G_{k+1} \trianglelefteq G$ and for all $i < j \le k$, $w_{i,i} = g_{i+1}^{d_{i,i}} v_{i,i}$ with $0 \le d_{i,i} < e_{i+1}$ and $w_{i,j} = g_j v_{i,j}$, where the $v_{i,i}$ and $v_{i,j}$ are reduced words in G_{k+1}. Let $w \in G$ be a reduced word.*

(a) *If $1 \le i \le k < n$ and $0 \le b < e_i$, then the word wg_i^b can be reduced to normal form using*

$$O(l_i(n-i)L_{k+1} + (L_{i+1} - L_{k+1})(n-k)L_{k+1})$$

multiplications in G_{k+1} and requiring $O((n-i)L_{k+1})$ bits of workspace.

(b) *If $k < n$, a multiplication and an inversion in G can be carried out using*

$$O(k(n-k)(L_1 - L_{k+1})L_{k+1})$$

multiplications in G_{k+1}, requiring $O(nL_{k+1})$ bits of workspace.

(c) *If $k = n$, then a multiplication or an inversion in G requires at most*

$$O(nL_1)$$

bit operations and $O(1)$ bits of workspace.

Proof. Let "multiplication" mean "multiplication in G_{k+1}".

(a) An easy induction argument shows that

$$g_j^{g_i^b} = g_j v_{i,j} v_{i,j}^{g_i} v_{i,j}^{g_i^2} \cdots v_{i,j}^{g_i^{b-1}}.$$

Therefore, by Lemma 3.2, reduced words $u_{i+1}, \dots, u_k \in G_{k+1}$ such that $g_j^{g_i^b} = g_j u_j$ can be computed using at most $O(l_i(n-i)L_{k+1})$ multiplications. Now assume that $g_1^{a_1} \dots g_n^{a_n}$ is a reduced word in G. Then

$$\begin{aligned} g_1^{a_1} \cdots g_n^{a_n} g_i^b &= g_1^{a_1} \dots g_{i-1}^{a_{i-1}} g_i^{a_i} g_i^b (g_{i+1}^{g_i^b})^{a_{i+1}} \dots (g_k^{g_i})^{a_k} (g_{k+1} \cdots g_n)^{g_i^b} \\ &= g_1^{a_1} \dots g_{i-1}^{a_{i-1}} g_i^{a_i} g_i^b (g_{i+1} u_{i+1})^{a_{i+1}} \dots (g_k u_k)^{a_k} (g_{k+1} \cdots g_n)^{g_i^b} \\ &= g_1^{a_1} \dots g_{i-1}^{a_{i-1}} g_i^{c_i} g_{i+1}^{d_{i+1}} v_{i+1} (g_{i+1} u_{i+1})^{a_{i+1}} \dots (g_k u_k)^{a_k} (g_{k+1} \cdots g_n)^{g_i^b} \end{aligned}$$

where either $c = a_i + b < e_i$, $d_{i+1} = 0$ and $v_{i+1} = 1$, or $a_i + b \geq e_i$, $c = a + b_i - e_i$, $d_{i+1} = d_{i,i}$, and $v_{i+1} = v_{i,i}$.

Let $i \leq j \leq k$, $0 \leq a, c < e_j$ and $u, v \in G_{k+1}$. Since

$$g_j^c v(g_j u)^a = g_j^{a+c} v^{g_j^a} u^{g_j^{a-1}} u^{g_j^{a-2}} \cdots u,$$

by Lemma 3.2, we obtain a reduced word $w \in G_{k+1}$ such that $g_j^c u(g_j v)^a = g_j^{a+c} w$ using $O(l_j(n-k)L_{k+1})$ multiplications. This also covers $O(l_j)$ bit operations required for the addition of a and c. Thus, for $j = i+1, \dots, k-1$, we use this to replace $g_j^{d_j} v_j (g_j u_j)^{a_j}$ by a reduced expression of the form $g_j^{c_j} g_{j+1}^{d_{j+1}} v_{j+1}$, where $0 \leq c_j < e_j$, $0 \leq d_{j+1} < e_{j+1}$ and $v_{j+1} \in G_{k+1}$. Therefore, at a cost of $O((L_{i+1} - L_{k+1})(n-k)L_{k+1})$ multiplications, we obtain

$$g_1^{a_1} \dots g_n^{a_n} g_i^b = g_1^{a_1} \dots g_{i-1}^{a_{i-1}} g_i^{c_i} \dots g_{k-1}^{c_{k-1}} g_k^{d_k} v_k (g_{k+1} \cdots g_n)^{g_i^b},$$

which can be reduced to normal form using $O(l_i(n-k)L_{k+1})$ multiplications. The claim now follows.

(b) Let $g = g_1^{a_1} \dots g_n^{a_n}$ and $h = g_1^{b_1} \dots g_n^{b_n}$ be reduced words. Successively multiplying the first word by $g_1^{b_1}, \dots g_n^{b_n}$ as in (a) and then by the reduced word $g_{k+1}^{b_{k+1}} \dots g_n^{b_n} \in G_{k+1}$ gives the normal form of the product. This requires

$$O((L_1 - L_{k+1})nL_{k+1} + k(n-k)(L_1 - L_{k+1})L_{k+1})$$

multiplications, which proves (b). To invert g, we choose h by setting $b_1 \equiv -a_1 \pmod{e_1}$, $b_i \equiv -a_i - d_{i-1,i-1}$ for $i = 2, \dots, k$ and $b_{k+1} = \dots = b_n = 0$. If w is the reduced form of gh, then $w \in G_{k+1}$, and the normal form v of its inverse can be found at a cost of $O((n-k)L_{k+1})$ multiplications by Lemma 3.3. Clearly, $g_1^{b_1} \dots g_k^{b_k} v$ is a reduced expression for the inverse of g.

(c) Note that this implies that G is abelian. Thus, an argument similar to (a) shows that wg_i^b can be reduced to its normal form using $O(L)$ multiplications in G. The result now follows as in (b). □

3.5 Remark. Instead of re-computing the x_l in Algorithm 3.1 and related algorithms in Lemma 3.2 each time, one may store them, once they are computed for the first time. This requires $O(nL_1^2)$ bits of storage. If all necessary x_j are known, this reduces the cost in Lemma 3.2 and Proposition 3.4 (a) and (b) to $O(mrL_{k+1})$, $O(l_i k + (L_{i+1} - L_{k+1})L_{k+1})$, and $O(k(L_1 - L_{k+1})L_{k+1})$, respectively. .

4. Rewriting words with respect to polycyclic presentations

Let $(h_1, \dots, h_r)$ be a sequence of elements of G and for $i = 1, \dots, r+1$, write $H_i = \langle h_i, \dots, h_r \rangle$. The sequence $(h_1, \dots, h_r)$ is a *polycyclic generating sequence* of a subgroup H of G if $H = H_1$ and for $i = 1, \dots, r$, H_{i+1} is a proper normal subgroup of H_i such that H_i/H_{i+1} is cyclic. The series

$$H = H_1 \rhd H_2 \rhd \cdots \rhd H_r \rhd H_{r+1} = 1$$

is the polycyclic series of H *associated with* $(h_1, \dots, h_r)$.

Evidently, $(g_1, \dots, g_n)$ is a polycyclic generating sequence of G. Conversely, if $(h_1, \dots, h_r)$ is a polycyclic generating sequence of H, then there exists a polycyclic presentation of H with generators $h_1, \dots, h_r$.

Let $g = g_1^{a_1} \dots g_n^{a_n}$ with $0 \le a_i < e_i$. If $g \ne 1$, the *depth* $\mathrm{dp}(h)$ of $h \ne 1$ is the least integer i such that $a_i \ne 0$, the *leading coefficient* $\mathrm{lc}(g)$ is a_i and the *relative order* $\mathrm{ro}(g)$ is the integer e_i. We put $\mathrm{dp}(1) = n + 1$ and $\mathrm{lc}(1) = 0$.

A polycyclic generating sequence $(h_1, \dots, h_r)$ of H is an *induced generating sequence* of H with respect to $\mathscr{P}$ if $\mathrm{dp}(h_i) < \mathrm{dp}(h_{i+1}$ for $i = 1, \dots, r$.

Let H be a subgroup of G with polycyclic generating sequence $h_1, \dots, h_r$. The following two algorithms serve to rewrite a reduced word $w \in H$ in the generators $g_1, \dots, g_n$ as a reduced word in $h_1, \dots, h_r$, and may also be used as a membership test for H. For the case when the e_i are primes, very similar algorithms are implemented in the computer algebra system GAP [3], but seem not to have been published.

4.1 Algorithm.

Input: a polycyclic presentation $\mathscr{P}$ of G, a polycyclic generating sequence $h_1, \dots, h_r$ of a subgroup H of G, and an element $g \in G$, where $h_1, \dots, h_r$ and g are given as normal words in $g_1, \dots, g_n$.

Output: integers $c_1, \dots, c_r$, $d_1, \dots, d_r$, $j_1, \dots, j_n$ and normal words $h_1^*, \dots, h_r^* \in G$ such that

(a) $H_{i+1}h_i^* = H_{i+1}h_i^{c_i}$ and $\langle H_{i+1}h_i^* \rangle = \langle H_{i+1}h_i \rangle$;

(b) $(h_i^*)^{d_i} \in H_{i+1}$, and if $1 \le k < d_i$ and $x \in H_{i+1}(h_i^*)^k$, then $\mathrm{dp}(x) \le \mathrm{dp}(h_i^*)$ and $|H_i : H_{i+1}| = d_i$;

(c) if $h_i^* \ne 1$, then $\mathrm{lc}(h_i^*)$ divides $\mathrm{ro}(h_i^*)$, and if $i < j$ and $\mathrm{dp}(h_i^*) = \mathrm{dp}(h_j^*)$, then $\mathrm{lc}(h_i^*)$ divides $\mathrm{lc}(h_j^*)$,

(d) j_l is the least integer such that $\mathrm{dp}(h_{j_l}^*) = l$ if such an integer exists, and $j_l = n+1$ otherwise.

where $H_i = \langle h_i, \dots, h_r \rangle$ for $i = 1, \dots, r + 1$.

```
(1) for $d := 1, \dots, n$ do let $j_d := n + 1$;
(2) for $i := r, r-1, \dots, 1$ do
(3)     let $h_i^* := h_i$;
(4)     let $d := \mathrm{dp}(h_i^*)$; let $j := j_d$;
(5)     if $j \le r$ then let $e := \mathrm{lc}(h_j^*)$, $h := h_j^*$ else let $e := \mathrm{ro}(h_i^*)$, $h := 1$;
(6)     call the Extended Euclidean Algorithm to compute $0 \le c_i, t < \mathrm{ro}(h_i^*)$ such
            that $c_i \,\mathrm{lc}(h_i^*) + te \equiv \gcd(\mathrm{lc}(h_i^*), e) \pmod{\mathrm{ro}(h_i^*)}$; let $h_i^* := h^t (h_i^*)^{c_i}$;
(7)     if $h_i^* \ne 1$ and $\mathrm{lc}(h_i^*) = \mathrm{lc}(h)$, then let $h_i^* := h^{-1}h_i^*$; go to (4);
(8)     if $h_i^* \ne 1$ then let $d_i = e/\mathrm{lc}(h_i^*)$; let $j_d := i$; else let $d_i = 1$;
```

4.2 Proposition. *Algorithm 4.1 is correct and requires $O(rL)$ multiplications in G and $O(rI)$ bit operations. Moreover, the sequence $(h_{j_i})_{j_i \le n}$ is an induced generating sequence of H.*

Proof. If the condition in Step (7) is true, $d = \operatorname{dp}(h_i^*)$ increases, so that for each i, this can only happen n times. The Extended Euclidean Algorithm requires

$$O(\log^3 \operatorname{ro}(h_i^*)) = O(l_d^3)$$

bit operations, and computing the normal form of $h^t (h_i^*)^{c_i}$ needs $O(l_d)$ multiplications in G. This gives the above bounds on the complexity. By induction on $r - i$, it is easy to see that properties (a) – (d) of Algorithm 4.1 hold. □

The following algorithm uses the output of Algorithm 4.1 to rewrite a reduced word in $g_1, \ldots, g_n$ as a reduced word in an arbitrary polycyclic generating sequence of a subgroup of G.

4.3 Algorithm (Exponents with respect to polycyclic generating sequence).
Input: a polycyclic presentation $\mathscr{P}$ of G, a polycyclic generating sequence $h_1, \ldots, h_r$ of a subgroup H of G, $h_1^*, \ldots, h_r^* \in G$, integers $c_1, \ldots, c_r$, $d_1, \ldots, d_r$, $j_1, \ldots, j_n$ satisfying properties (a) – (d) of Algorithm 4.1, and an element $g \in G$, where $h_1, \ldots, h_r$ and g are given as normal words in $g_1, \ldots, g_n$.
Output: integers $a_1, \ldots, a_r$ and $w \in G$ such that $g = h_1^{a_1} \ldots h_r^{a_r} w$ satisfying $0 \le a_i < d_i = |H_i : H_{i+1}|$, and $w = 1$ if $g \in H$, where $H_i = \langle h_i, \ldots, h_r \rangle$ for $i = 1, \ldots, r+1$.

```
(1) for $i = 1, \ldots, r$ do
(2)     let $v := w$; let $d := \operatorname{dp}(v)$;
(3)     while $d \le n$ and $i < j_d \le n$ and $\operatorname{lc}(h_{j_d}^*)$ divides $\operatorname{lc}(v)$ do
(4)         let $c := e_d - \operatorname{lc}(v)/\operatorname{lc}(h_{j_d}^*)$; let $v := (h_{j_d}^*)^c v$; let $d := \operatorname{dp}(v)$;
(5)     if $v \ne 1$, then
(6)         if $d = \operatorname{dp}(h_i^*)$ and $\operatorname{lc}(h_i^*)$ divides $\operatorname{lc}(v)$, then
(7)             let $a_i = c_i \operatorname{lc}(v)/\operatorname{lc}(h_i^*) \bmod d_i$; let $w := h_i^{-a_i} w$;
(8)         else stop.
```

4.4 Proposition. *Algorithm 4.3 is correct and requires $O(rL)$ multiplications in G.*

Proof. Note first that for each i, Step (4) of Algorithm 4.3 is executed at most n times, since $\operatorname{dp}(h_i^*)$ increases. Since $c < e_d$, computing $(h_{j_d}^*)^c v$ requires at most $O(l_d)$ multiplications, giving a cost of $O(L)$ for each i. This also covers the cost of $O(n + l_d)$ for computing $h_i^{-a_i} w$.

It is also clear that $g = h_1^{a_1} \ldots h_r^{a_r} w$ with $0 \le a_i < d_i$. Therefore it remains to consider the case when $w \in H$. Now assume inductively that $w \in H_i$ at Step (2). Then at Step (5), v is an element of maximal depth in the coset $H_{i+1} v = H_{i+1} w$. In particular, $w \in H_{i+1}$ if $v = 1$.

If $v \neq 1$, by properties (a) and (b) of Algorithm 4.1, there exists an integer k with $0 \leq k < d_i$ such that $H_{i+1}v = H_{i+1}w = H_{i+1}(h_i^*)^k = H_{i+1}h_i^{c_i k}$. By the minimality of $\mathrm{dp}(v)$ and $\mathrm{dp}(h_i^*)$, we have $\mathrm{dp}(v) = \mathrm{dp}(h_i^*)$, and so $\mathrm{lc}(v) \equiv k\,\mathrm{lc}(h_i^*) \pmod{\mathrm{ro}(h_i^*)}$. Since $\mathrm{lc}(h_i^*)$ divides $\mathrm{ro}(h_i^*)$, it follows that $\mathrm{lc}(v)$ is divisible by $\mathrm{lc}(h_i^*)$ and $k = \mathrm{lc}(v)/\,\mathrm{lc}(h_i^*)$. Thus, we have $H_{i+1}w = H_{i+1}h_i^{a_i}$, and so $h_i^{-a_i}w \in H_{i+1}$, as required. □

Algorithm 4.1 and Algorithm 4.3 require that the generators of the polycyclic generating sequence $(h_1, \ldots, h_r)$ are given as words in the generators of $\mathscr{P}$. The following result can sometimes be used if the g_i are given as reduced words in the h_i.

4.5 Proposition. *Let $(h_1, \ldots, h_n)$ be a polycyclic generating sequence of G, and assume that $g_i = h_i h_{i+1}^{a_{i,i+1}} h_{i+2}^{a_{i,i+2}} \ldots, h_n^{a_{i,n}}$ is a reduced word in $h_1, \ldots, h_n$ for $i = 1, \ldots, n$.*

(a) *There exists an algorithm which, given the $a_{i,j}$ above, computes reduced expressions of $h_1, \ldots, h_n$ at a cost of $O(nL)$ multiplications in G.*

(b) *Given reduced expressions for the h_i as in* (a)*, a reduced word in $g_1, \ldots, g_n$ can be rewritten as a reduced word in $h_1, \ldots, h_n$ using $O(nL)$ multiplications in G. The opposite direction requires $O(L)$ multiplications.*

Proof. Observe first that $\langle h_i, \ldots, h_n \rangle = \langle g_i, \ldots, g_n \rangle$ for $i = 1, \ldots, n$, so that $(h_1, \ldots, h_n)$ is, in fact, an induced generating sequence of G.

We show that every h_i can be written as a reduced word v_i in $g_1, \ldots, g_n$. Assume that this is true for $h_{i+1}, \ldots, h_n$. Then $h_i = g_i(h_{i+1}^{a_{i,i+1}} \ldots, h_n^{a_{i,n}})^{-1}$ Replacing each h_j on the right hand side by a reduced expression in $g_{i+1}, \ldots, g_n$, a reduced expression for h_i in $g_i, \ldots, g_n$ can be obtained using $O(L_{i+1})$ multiplications in G, taking into account that the reduced form of an inverse in G_{i+1} requires $n - i \in O(L)$ multiplications by Lemma 3.3.

Therefore we obtain reduced expressions of the h_i using $O(nL)$ multiplications. Now every reduced word in $h_1, \ldots, h_n$ can be rewritten as a reduced word in $g_1, \ldots, g_n$ by substituting reduced expressions for the h_i and evaluating, at a cost of $O(L)$ multiplications in G. To rewrite a reduced word in $g_1, \ldots, g_n$, observe that we may choose $h_i^* = h_i$, $d_i = e_i$, $c_i = 1$ in Algorithm 4.3. Thus, Proposition 4.4 gives the result. □

Sometimes, a polycyclic presentation is changed by multiplying generators by elements of a small G_{k+1}. In this case, a new presentation can be computed using multiplications in G_{k+1} alone.

4.6 Proposition. *Assume that $G_{k+1} \lhd G$ and let $(h_1, \ldots, h_n)$ be a polycyclic generating sequence of G such that $h_i = g_i u_i$ for all $i = 1, \ldots, k$, where $u_1, \ldots, u_k$ are reduced words in G_{k+1}, and $h_i \in G_{k+1}$ for $i = k+1, \ldots, n$. Then the following statements hold.*

(a) *A reduced word in $h_1, \ldots, h_n$ can be rewritten as a reduced word in $g_1, \ldots, g_n$ using $O(k(L_1 - L_{k+1})L_{k+1})$ multiplications in G_{k+1}.*

(b) *A reduced word in $g_1, \ldots, g_n$ be rewritten as a reduced word in $h_1, \ldots, h_n$ using $O(n(L_1 - L_{k+1})L_{k+1})$ multiplications in G_{k+1} and $O((n-k)I_{k+1})$ bit operations.*

(c) *A polycyclic presentation of G with generators $h_1, \dots, h_n$ can be computed using $O(n^3(L_1 - L_{k+1})L_{k+1})$ multiplications in G_{k+1} and $O((n-k)I_{k+1})$ bit operations.*

Proof. Let "multiplications" mean "multiplications in G_{k+1}".

(a) Let $h = h_1^{a_1} \dots h_n^{a_n}$ be a reduced word in $h_1, \dots, h_n$ and $u \in G_{k+1}$ be reduced words. If $i \le k$, we have

$$uh_i^{a_i} = u(g_iu_i)^{a_i} = g_i^{a_i}u^{g_i^{a_i}}u_i^{g_i^{a_i-1}}u_i^{g_i^{a_i-2}}\dots u_i^{g_i}u_i, \qquad (*)$$

so that by Lemma 3.2, a word $v \in G_{k+1}$ such that $g_i^{a_i}v$ is the normal form of $uh_i^{a_i}$ can be computed using at most $O(nl_iL_{k+1})$ multiplications in G_{k+1}. Thus, the word $h = h_1^{a_1}\dots h_k^{a_k}$ can be rewritten as a reduced word $g_1^{a_1}\dots g_k^{a_kw}$ with $w \in G_{k+1}$. in $g_1, \dots, g_n$ at a cost of $O(n(L_1 - L_{k+1})L_{k+1})$ multiplications in G_{k+1}. Substituting reduced words for $h_{k+1}, \dots, h_n$, the word $wh_{k+1}^{a_{k+1}}\dots h_n^{a_n}$ can be written as a reduced word in $g_{k+1}, \dots, g_n$ using L_{k+1} multiplications.

(b) Let $g = g_1^{a_1}\dots g_n^{a_n}$ be a reduced word, and for $i \le k$, write $h_i^{a_i} = g_i^{a_i}v_i$, where $v_i \in G_{k+1}$, taking $u = 1$ in $(*)$. Then

$$h_1^{-a_1}g = v_i^{-1}g_2^{a_2}\dots g_n^{a_n} = g_2^{a_2}\dots g_k^{a_k}(v_i^{-1})^{g_2^{a_2}\dots g_k^{a_k}}g_{k+1}^{a_{k+1}}\dots g_n^{a_n}.$$

Continuing like this, we obtain that

$$\begin{aligned} h_k^{-a_k}\dots h_1^{-a_1}g &= v_k^{-1}(v_{k-1}^{-1})^{g_k^{a_k}}\dots(v_2^{-1})^{g_3^{a_3}\dots g_k^{a_k}}(v_1^{-1})^{g_2^{a_2}\dots g_k^{a_k}}g_{k+1}^{a_{k+1}}\dots g_n^{a_n} \\ &= v_k^{-1}(v_{k-1}^{-1}(\dots(v_2^{-1}(v_1^{-1})^{g_2^{a_2}})^{g_3^{a_3}}\dots)^{g_{k-1}^{a_{k-1}}})^{g_k^{a_k}}g_{k+1}^{a_{k+1}}\dots g_n^{a_n} \in G_{k+1}. \end{aligned}$$

Call the last expression v. Inversion in G_{k+1} requires at most $n - k$ multiplications by Lemma 3.3, while by Lemma 3.2, conjugation by a $g_i^{a_i}$ requires at most $O(l_inL_{k+1})$ multiplications. Thus, v can be reduced to normal form in $g_{k+1}, \dots, g_n$ using $O(n(L_1 - L_{k+1})L_{k+1})$ multiplications. Since $(h_{k+1}, \dots, h_n)$ is a polycyclic generating system of G_{k+1}, by Proposition 4.2 and Proposition 4.4, this word can be rewritten as a normal word in $h_{k+1}, \dots, h_n$ using $O((n-k)L_{k+1})$ multiplications and $O((n-k)I_{k+1})$ bit operations.

(c) Let $1 \le i \le k$, then

$$h_i^p = (g_iu_i)^p = w_{i,i}u_i^{g_i^{p-1}}u_i^{g_i^{p-2}}\dots u_i^{g_i}u_i.$$

By Lemma 3.2, a reduced expression for h_i^p in $g_1, \dots, g_n$ can be computed using $O(nl_iL_{k+1})$ multiplications. If $1 \le i < j \le k$, then

$$h_jh_i = g_jg_iu_j^{g_i}u_i = g_iw_{i,j}u_j^{g_i}u_i,$$

and by Lemma 3.2, we obtain a reduced expression in $g_1, \dots, g_n$ of the right hand side using $O(L_{k+1})$ multiplications. In both cases, the right hand side can then be rewritten as a reduced word in the h_i using $O(nL_1L_{k+1})$ multiplications by (a). □

5. A subexponential collection algorithm

Let d denote the derived length of G. The polycyclic presentation $\mathscr{P}$ of G *exhibits the cycle structure of the derived series of* G if there exist integers $i_1, \ldots, i_d$ such that, for $k = 1, \ldots, d$, $G^{(k)} = G_{i_k}$, and

$$G^{(k-1)}/G^{(k)} = \langle\, h_{i_k} G^{(k)} \,\rangle \times \langle\, h_{i_k+1} G^{(k)} \,\rangle \times \cdots \times \langle\, h_{i_{k+1}} G^{(k)} \,\rangle.$$

Let $\mathbf{M}_{\mathrm{der}}(G)$ and $\mathbf{W}_{\mathrm{der}}$ denote the maximum number of bit operations and bits of workspace required to perform a multiplication in G if $\mathscr{P}$ exhibits the cycle structure of the derived series of G.

5.1 Theorem. *Let d and l denote the derived and composition lengths of G, respectively.*

(a)

$$\mathbf{M}_{\mathrm{der}}(G) \leq (ClL)^{2d-1} \leq e^{C' \log l \log L},$$

where C and C' are suitable constants, and $\mathbf{W}_{\mathrm{der}} \in O(dL)$.

(b) *A polycyclic presentation $\mathscr{P}'$ exhibiting the cycle structure of the derived series of G can be computed using $O(l^4 L^3 \mathbf{M}_{\mathrm{der}}(G) + dlL^6)$ bit operations.*

(c) *Any reduced word with respect to $\mathscr{P}$ can be rewritten as a reduced word with respect to $\mathscr{P}'$ using $O(L\mathbf{M}_{\mathrm{der}}(G))$ bit operations; the other direction requires $O(nL\mathbf{M}_{\mathrm{der}}(G) + lI)$ bit operations.*

Proof. (a) Let $C \in \mathbb{R}$ be such that, for all G and $\mathscr{P}$ exhibiting the cycle structure of the derived series of G, a multiplication in G can be carried out using at most $(CL)^4$ multiplications in G' if G is not abelian, and using at most ClL bit operations if G is abelian. Such a C exists by (b) and (c) of Proposition 3.4. Induction on d then shows that a multiplication requires $(ClL)^{2d-1}$ bit operations. Note that by [6], we have $d \in O(\log l)$, from which the second part of the inequality follows.

(b) Assume that $1 \leq k \leq n$ such that $\mathscr{P}_{k+1}$ is a polycyclic presentation of G_{k+1} with generators $h_1, \ldots, h_r$ such that $(h_1, \ldots, h_r)$ exhibits the cycle structure of the derived series of G_{k+1}, and that, for $k < i \leq n$, g_k can be expressed as a reduced word in $h_1, \ldots, h_r$. Let “multiplication”, “induced generating sequence”, “normal form” all refer to $\mathscr{P}_{k+1}$, unless specified otherwise.

Using Algorithm 4.1 and Algorithm 4.3, each h_j can be rewritten as a word $g_{k+1}^{a_{k+1}} \cdots g_n^{a_n}$, using $O(lL_{k+1})$ multiplications and $O(lI_{k+1})$ bit operations. Now

$$h_j g_k = g_k \underbrace{w_{k,k+1}^{a_{k+1}} \cdots w_{k,n}^{a_n}}_{v},$$

where v is a word in $g_{k+1}, \ldots, g_n$. By substituting words in $h_1, \ldots, h_r$ for $g_{k+1}, \ldots, g_n$ in the $w_{k,j}$ and multiplying, the words $w_{k,j}$ can be rewritten as words in $h_1, \ldots, h_r$, from which we obtain a reduced expression of v at a cost of $O(L_{k+1}^2)$. In a similar way, we obtain a reduced expression of $g_k^{e_k} = w_{k,k}$. Putting $h_0 = g_k$, this yields a polycyclic presentation $\mathscr{P}_k^*$ of G_k with generators $h_0, \dots, h_r$

Let $g \in G_{k+1}$, then the normal form of g^{g_k}, and hence of a commutator $[g, g_k] = g^{-1}g^{g_k}$ can be computed using $O(L_{k+1})$ multiplications by Lemma 3.3 and Lemma 3.2. The normal forms of a commutator $[g, h]$ with $h \in G_{k+1}$ and g^{e_i} require $O(l)$ and $O(l_i)$ multiplications, respectively. This shows that an induced generating sequence $(t_1, \dots, t_s)$ for G_k' can be computed using $O(l^2 L_{k+1})$ multiplications and $O(l^2 I_{k+1})$ bit operations.

Next, apply Algorithm 4.1 to the sequence $(h_1, \dots, h_r, t_1, \dots, t_s)$ to obtain $f_1, \dots, f_r$, defined by $f_i = |\langle h_i, \dots, h_r, t_1, \dots, t_s \rangle : \langle h_{i+1}, \dots, h_r, t_1, \dots, t_s \rangle|$. This requires $O(lL_{k+1})$ multiplications and $O(lI_{k+1})$ bit operations. In addition, let $f_0 = e_k$ and put $M = 2f_0 \cdots f_r$. Then $h_i^{f_i} = h_0^{a_{i,0}} \cdot \dots \cdot h_r^{a_{i,r}} G_k'$ for $i = 0, \dots, r$. Let $B = (b_{i,j})_{0 \le i,j \le r}$, where $b_{i,j} = a_{i,j}$ if $i \ne j$, and let $b_{i,i} = (a_{i,i} - f_i) \bmod M$. Using a standard algorithm for computing the Smith normal form of B modulo M (see, e. g., [11, Section 8.3]), we compute integral matrices P, D, Q with entries in the range $\{0, \dots, M-1\}$ such that $\det(P) \equiv \det(Q) \equiv 1 \pmod{M}$ and D is a diagonal matrix (but not necessarily with entries dividing each other) with entries $d_1, \dots, d_r$, such that $B \equiv PDQ \pmod{M}$, at a cost of $O(r^3 \log^3 M + r^2 \log^4 M) \subseteq O(L_k^6)$ bit operations. If $Q = (q_{i,j})_{0 \le i,j \le r}$ and $x_i = h_0^{q_{i,0}} \cdot \dots \cdot h_r^{q_{i,r}}$, then it follows that $x_i^{d_i} \in G_k'$ for $i = 0, \dots, r$, and, after removing those x_i with $d_i = 1$, the sequence $(x_0, \dots, x_r, t_1, \dots, t_s)$ is a polycyclic generating sequence of G_k such that

$$G_k/G_k' = \langle x_0 G_k' \rangle \times \dots \times \langle x_r G_k' \rangle.$$

In a similar way, we can replace $t_1, \dots, t_s$ by a polycyclic generating sequence of G_k' exhibiting the cycle structure of G_k'. This yields a polycyclic generating sequence $y_1, \dots, y_t$ of G_k exhibiting the cycle structure of G_k,where each y_i is given as a reduced word in $h_0, \dots, h_r$. Now Proposition 3.4 shows that a multiplication with respect to $\mathscr{P}_k^*$ of two reduced words in the generators $h_0, \dots, h_r$ can be performed at a cost of $O(l_k l L_{k+1})$ multiplications with respect to $\mathscr{P}_{k+1}$. This allows to compute $y_j y_i$ and y_i^e as reduced words in $h_0, \dots, h_r$, where $e = |\langle y_i, \dots, y_t \rangle : \langle y_{i+1}, \dots, y_t \rangle|$. Using Algorithm 4.1 and Algorithm 4.3, a reduced word in $h_0, \dots, h_r$, can be rewritten in the generators $y_1, \dots, y_t$, at a cost of $O(l^2 l_k L_k^2)$ multiplications and $O(lI)$ bit operations. In this way, a presentation of G_k with generators $y_1, \dots, y_t$, and each g_i (which can be written as a word in $h_0, \dots, h_r$) can be written as a word in $y_1, \dots, y_t$. Thus, using $O(l^4 L^3)$ multiplications and $O(dL^6)$ bit operations, this allows to replace k by $k-1$ in our initial assumption, and we may repeat the above procedure if $k > 1$. We let $\mathscr{P}' = \mathscr{P}_1$.

(c) Let $h_1, \dots, h_r$ be the generators of $\mathscr{P}'$. Substituting reduced expressions in $h_1, \dots, h_r$ for $g_1, \dots, g_n$ and multiplying with respect to $\mathscr{P}'$ yields a reduced word in $h_1, \dots, h_r$, using $O(L)$ such multiplications. By Algorithm 4.1 and Algorithm 4.3, any reduced expression in $h_1, \dots, h_r$ can be rewritten as a reduced expression in $g_1, \dots, g_n$ using $O(lL)$ multiplications with respect to $\mathscr{P}'$ and $O(lI)$ bit operations. □

6. Reduction to elementary abelian normal series

A polycyclic generating sequence $(h_1, \ldots, h_n)$ of G *refines* a series

$$H = H_1 \triangleright H_2 \triangleright \cdots \triangleright H_r = 1$$

of the subgroup H of G if there exist $j_1, \ldots, j_r$ such that $H_i = \langle h_{j_i}, \ldots, h_n \rangle$ for $i = 1, \ldots, r$.

In this section, we reduce the problem of multiplying with respect to $\mathscr{P}$ to the case when $\mathscr{P}$ refines an elementary abelian normal series. This implies that the e_i are all primes. We proceed in two steps, first showing that we may assume that the e_i are primes. Note that these results can also be deduced from Theorem 5.1, albeit with a significantly worse bound on the cost.

Assume that $\mathbf{P}$, $\mathbf{M}$, $\mathbf{P}_{comp}$, $\mathbf{M}_{comp}$, $\mathbf{P}_{\mathrm{elab}}$ and $\mathbf{M}_{\mathrm{elab}}$ are functions such that the maximum number of bit operations required for r multiplications in G is in $O(\mathbf{P}(G) + r\mathbf{M}(G))$ for any $\mathscr{P}$, that it is in $O(\mathbf{P}_{\mathrm{comp}}(G) + r\mathbf{M}_{\mathrm{comp}}(G))$ when the G_i form a composition series of G, and in $O(\mathbf{P}_{\mathrm{elab}}(G) + r\mathbf{M}_{\mathrm{elab}}(G))$ when the G_i refine an elementary abelian series of G.

6.1 Proposition. *Assume that for each e_i, the prime factorisation is given. Then a polycyclic presentation $\mathscr{P}'$ whose associated polycyclic series is a composition series of G and such that the generators of $\mathscr{P}'$ are powers of those of $\mathscr{P}$ can be computed using $O(n\mathbf{P}_{\mathrm{comp}}(G) + nL^2\mathbf{M}_{\mathrm{comp}}(G) + n^2 I)$ bit operations. Moreover, a reduced word with respect to $\mathscr{P}$ can be converted into a reduced word with respect to $\mathscr{P}'$ and vice versa using $O(I)$ bit operations. Thus, we may choose $\mathbf{P}(G) = n\mathbf{P}_{\mathrm{comp}}(G) + nL^2\mathbf{M}_{\mathrm{comp}}(G) + n^2 I$ and $\mathbf{M}(G) = I + \mathbf{M}_{\mathrm{comp}}(G)$.*

Proof. Let $\mathscr{P}_{n+1}$ be the empty presentation, and assume that we have defined a polycyclic presentation $\mathscr{P}_{k+1}$ of G_{k+1} whose associated series is a composition series of G_{k+1}s. such that the generators of $\mathscr{P}_{k+1}$ are powers of $g_{k+1}, \ldots, g_n$. Let $e_k = p_{k,1} \cdots p_{k,r}$ be a prime factorisation of e_k. As generators for $\mathscr{P}_k$, we choose

$$h_1 = g_k, h_2 = g_k^{p_{k,1}}, h_3 = g_k^{p_{k,1}p_{k,2}, \ldots, h_r} = g_k^{p_{k,1} \cdots p_{k,r-1}},$$

together with the generators $h_{r+1}, \ldots, h_s$ of $\mathscr{P}_{k+1}$. If $0 \le a < e_k$, write

$$a = a_1 + p_{k,1}(a_2 + p_{k,2}(a_3 + \ldots))$$

with $0 \le a_i < p_{k,i}$ for $i = 1, \ldots, r$, at a cost of $O(rl_k^2) \subseteq O(l_k^3)$ bit operations. Moreover, a can be computed from the a_i at the same cost. Therefore, a reduced word in generators $h_{r+1}, \ldots, h_s$ can be converted into a reduced word in generators $g_k, \ldots, g_n$ and back at a cost of $O(I_k)$ bit operations.

To obtain the relations of $\mathscr{P}_k$, we take the set of relations of $\mathscr{P}_{k+1}$ and add relations $h_j h_i = h_i h_j$, $1 \le i < j < r$, $h_i^{p_{k,i}} = h_{i+1}$ if $i < r$, and for $1 \le i \le r < j \le s$, we add relations $h_j h_i = h_i v_{i,j}$, where $v_{r,r}$ and the $v_{i,j}$ are reduced words in $h_{r+1}, \ldots, h_s$

obtained as follows. Firstly, $v_{r,r}$ is $w_{k,k}$, rewritten as a reduced word in $h_{r+1}, \dots, h_s$, and if $r < j \le s$ and $h_j = g_m^a$ with $0 \le a < e_m$, we obtain $v_{1,j}$ by rewriting $w_{k,m}$ as a reduced word in $h_{r+1}, \dots, h_s$ and taking its ath power. If $1 \le i < r$ and $v_{i,j}$ is known for all j, then Lemma 3.2 can be used to compute

$$v_{i+1,j} = h_j^{h_i^{p_{k,i}}}$$

for all j at a cost of $O(\log p_{k,i}(n-k)L_{k+1})$ multiplications with respect to $\mathscr{P}_{k+1}$. Thus, at a total cost of $O(l_k(n-k)L_{k+1}$ such multiplications and $O((n-k)I_k)$ bit operations, we obtain $\mathscr{P}_k$ from $\mathscr{P}_{k+1}$. The total cost of computing $\mathscr{P}' = \mathscr{P}_1$ can thus be bounded by $O(nL^2\mathbf{M}_{\mathrm{comp}} + n^2 I)$ bit operations. Consequently, a multiplication with respect to $\mathscr{P}$ can be carried out by computing $\mathscr{P}'$, translating the words into reduced words with respect to $\mathscr{P}'$, performing the multiplication with respect to $\mathscr{P}'$, and translating the result back into a reduced word with respect to $\mathscr{P}$. □

Now we can prove the main result of this section. The proof is similar to that of Theorem 5.1 but gives a sharper bound on the cost involved.

6.2 Theorem. *Assume that $e_1, \dots, e_n$ are primes. Then a polycyclic presentation $\mathscr{P}'$ for G refining a normal series of G with elementary abelian factors can be computed using $O(n\mathbf{P}_{\mathrm{elab}} + n^4 L\mathbf{M}_{\mathrm{elab}}(G) + n^4 I)$ bit operations. Converting r reduced words in $g_1, \dots, g_n$ into reduced words in the generators of $\mathscr{P}'$ requires $O(\mathbf{P}_{\mathrm{elab}}(G) + rL\mathbf{M}_{\mathrm{elab}}(G))$ bit operations, and the other direction requires $O(\mathbf{P}_{\mathrm{elab}}(G) + rnL\mathbf{M}_{\mathrm{elab}}(G) + nI)$ bit operations. Therefore we may choose $\mathbf{P}_{\mathrm{comp}}(G) = n\mathbf{P}_{\mathrm{elab}} + n^4 L\mathbf{M}_{\mathrm{elab}}(G) + n^4 I$ and $\mathbf{M}_{\mathrm{comp}}(G) = nL\mathbf{M}_{\mathrm{elab}}(G)$.*

Proof. Suppose that $\mathscr{P}_{k+1}$ is a polycyclic presentation of G_{k+1} with generators $h_{k+1}, \dots, h_n$, and that $(h_{k+1}, \dots, h_n)$ refines a normal series with elementary abelian factors of G_{k+1}. Moreover, assume that we have computed reduced expressions in $h_{k+1}, \dots, h_n$ for $g_k, \dots, g_n$. Let "multiplication", "polycyclic generating sequence", "normal form" all refer to $\mathscr{P}_{k+1}$, unless specified otherwise.

Let $g, h \in G_{k+1}$, then the normal form of a commutator $[g, g_k] = g^{-1}g^{g_k}$ can be computed using $O(L_{k+1})$ multiplications by Lemma 3.3 and Lemma 3.2. The same holds for $[g, h]$, $g_k^{p_k}$ and g^{p_i}. Therefore an induced generating sequence $(t_1, \dots, t_r)$ for $H = G_k' G_k^{p_k}$ can be computed using $O((n-k)^2 L_{k+1})$ multiplications and $O((n-k)^2 I_{k+1})$ bit operations.

After deleting those h_i with $\mathrm{dp}(h_i) = \mathrm{dp}(t_j)$ for some j, the sequence

$$(g_k, h_{k+1}, \dots, h_n, t_1, \dots, t_r)$$

is a polycyclic generating sequence of G_k refining the series $G_k \rhd H \trianglerighteq 1$. If $H \neq 1$, then we obtain a polycyclic generating sequence of G refining the sequence $G_k \rhd H \rhd H'H^p \trianglerighteq 1$, where $p = \mathrm{ro}(x_l)$. Continuing like this, at a cost of $O((n-k)^3(L_{k+1}\mathbf{M}_{\mathrm{elab}}(G_{k+1}) + I_{k+1}))$, we obtain a polycyclic generating sequence $(g_k, y_{k+1}, \dots, y_n)$ refining a normal series of G_k with elementary abelian factors.

Now by Lemma 3.2, the normal form of $y_j^{g_k}$ can be computed using $O((n-k)L_{k+1})$ multiplications. To obtain the normal form of $g_k^{p_k}$, we substitute the normal forms of $g_{k+1}, \ldots, g_n$ into $w_{k,k}$ and multiply, at a cost of $O(L_{k+1})$ multiplications. Normal forms of $y_j^{y_i}$ and $y_j^{\mathrm{ro}(y_j)}$ can be computed in a straightforward way, using $O(n-k)$ and $O(\log \mathrm{ro}(y_j))$ multiplications, respectively. By Proposition 4.4, each of these normal words can be rewritten as a normal word in $y_{k+1}, \ldots, y_n$ using

$$O((n-k)^3 + (n-k)L_{k+1}) \subseteq O((n-k)^2 L_{k+1})$$

multiplications and $O((n-k)^3 + (n-k)I_{k+1})$ bit operations. The same holds for the normal forms of $g_{k+1}, \ldots, g_n$. Thus, a polycyclic presentation $\mathscr{P}_k$ of G_k with generators $g_k, y_{k+1}, \ldots, y_n$ can be computed using

$$O((n-k)^3(L_{k+1}\mathbf{M}_{\mathrm{elab}}(G_{k+1}) + I_{k+1}))$$

bit operations, thus allowing to replace k by $k-1$ in our initial assumption, and to repeat the above if $k > 0$.

The remaining statements follow as in the proof of Theorem 5.1. □

6.3 Remark. In a similar way, a polycyclic presentation refining the derived series of G can be computed using only operations in groups having such a polycyclic presentations, the number of such operations being polynomial in L_1. If G is nilpotent, then "derived series" may be replaced by "lower central series", or by "lower p-central series" if G is a p-group. The bounds on the cost remain the same in all cases.

7. Exhibited supplements

While the bounds obtained in Theorem 5.1 may be of some theoretical interest, they are usually far too bad to reflect empirical results. One of the main reasons seems to be that the multiplications in G_{k+1} used for the estimates in Proposition 3.4 and Theorem 5.1 do not actually take place in the whole of G_{k+1}, but in much smaller subgroups of G. This gives rise to the following definition.

Let $(h_1, \ldots, h_r)$ be a polycyclic generating sequence of G. Assume that H is a subgroup of G and let $\{\, i \mid h_i \in H \,\} = \{\, i_1, \ldots, i_s \,\}$, where $1 \leq i_1 < i_2 < \cdots < i_s \leq r$. The subgroup H of G is *exhibited* by $(h_1, \ldots, h_r)$ if $(h_{i_1}, \ldots, h_{i_s})$ is a polycyclic generating sequence of H. It is easy to see that the polycyclic presentation of H with generators $(h_{i_1}, \ldots, h_{i_s})$ is the restriction to H of the polycyclic presentation of G with generators $h_1, \ldots, h_r$. Obviously, G and the G_i are exhibited by $(g_1, \ldots, g_n)$.

Moreover, if H is an exhibited subgroup of G and $1 \leq i \leq r$, then the subgroups $H \cap < h_i, \ldots, h_r >$ is exhibited as well. More generally, it is not difficult to prove that intersections of subgroups exhibited by $(h_1, \ldots, h_r)$ are exhibited by $(h_1, \ldots, h_r)$.

The following result indicates the role that exhibited subgroups play in multiplication algorithms.

7.1 Proposition. *Assume that H is an exhibited subgroup of G, and that there exists $k \le n$ such that $G_{k+1} \trianglelefteq G$ with $G = HG_{k+1}$. Then a multiplication in G can be carried out using one multiplication in H and $O((L_1 - L_{k+1})(n-k)L_{k+1})$ multiplications in G_{k+1}.*

Proof. Let $g = g_1^{a_1} \dots g_n^{a_n}$, $h = g_1^{b_1} \dots g_n^{b_n}$ be reduced words in G and write

$$g_1^{a_1} \cdots g_n^{a_n} g_1^{b_1} \cdots g_n^{b_n} = \underbrace{g_1^{a_1} \cdots g_k^{a_k} g_1^{b_1} \cdots g_k^{b_k}}_{\in H} \underbrace{\left(g_{k+1}^{a_{k+1}} \cdots g_n^{a_n}\right)^{g_1^{b_1} \cdots g_k^{b_k}} g_{k+1}^{b_{k+1}} \cdots g_n^{b_n}}_{\in G_{k+1}}$$

Reducing the word $g_1^{a_1} \dots g_{i-1}^{a_{i-1}} g_1^{b_1} \dots g_{i-1}^{b_{i-1}}$ in H into normal form $g_1^{c_1} \dots g_n^{c_n}$ requires one multiplication in H.

By Lemma 3.2, the expression

$$\left(g_{k+1}^{c_{k+1}} \cdots g_n^{c_n}\right) \left(g_{k+1}^{a_{k+1}} \cdots g_n^{a_n}\right)^{g_1^{b_1} \cdots g_k^{b_k}} \left(g_{k+1}^{b_{k+1}} \cdots g_n^{b_n}\right)$$

can be reduced to its normal form $g_i^{c'_i} \cdots g_n^{c'_n}$ using at most $O((L_1 - L_k)(n-k+1)L_{k+1})$ multiplications in G_{k+1}. Clearly, $g_1^{c_1} \dots g_{i-1}^{c_{i-1}} g_i^{c'_i} \dots g_n^{c'_n}$ is a reduced expression for gh. □

7.2 Remark. The multiplication algorithm outlined in the proof of Proposition 7.1 is very similar to collection from the left if fast powering and fast conjugation are used. The main difference is that when collection from the left, a carry word $g_i^{c_i} \dots g_n^{c_n}$ occurs whenever a $g_i^{b_i}$ is moved across a word in G_{k+1}. However, the bound on the cost in Proposition 7.1 remains the same.

In order to exploit Proposition 7.1 to obtain an efficient multiplication algorithm, it is important to choose G_{k+1} in such a way that a multiplication in G_{k+1} is cheap. The following proposition illustrates the idea which we will be using. The N_i which we will choose are terms of a chief series, so that H will complement one of the factors.

7.3 Proposition. *Let $1 = N_0 \lhd N_1 \lhd \dots \lhd N_r = G$ be a normal series of G with nilpotent factors. Then there exist a subgroup H of G and $k \in \{1, \dots, r\}$ such that N_k is nilpotent and $G = HN_k$. Moreover, we also have $G = HK$ for some Sylow subgroup K of N_r.*

Proof. The statement certainly holds if $G = 1$. Therefore assume that $G \ne 1$ and let k be minimal such that $N_k \not\le \Phi(G)$. Then there exists a maximal subgroup H of G satisfying $HN_k = G$. Moreover, since $N_{k-1} \le \Phi(G)$ and N_k/N_{k-1} is nilpotent, also N_k is nilpotent; see, e. g. [2, A, 9.3 (c)]. Since $N_k \not\le \Phi(G)$, there exists a Sylow subgroup K of N_r such that $K \not\le \Phi(G)$. Clearly, $K \trianglelefteq G$ and $HK = G$. □

The following algorithm uses ideas very similar to Algorithm 3.1.

7.4 Algorithm PowerSum.

Input: an $n \times n$ matrix M over a ring R with multiplicative identity, an integer $k = \sum_{j=0}^{m-1} b_j 2^j$ with $b_j \in \{0, 1\}$

```
Output: $S = \sum_{i=0}^{k-1} M^i$
(1) let $S := 1_n$; $T := 1_n$;
(2) for $i = 0, \dots, m-1$ do
(3)     let $T := (T(M-1)+2)T$;
(4)     if $b_i = 1$, then $S := (T(M-1)+1)S + T$;
```

7.5 Lemma. *Algorithm 7.4 is correct and requires $O(m)$ multiplications and additions of $n \times n$ matrices over R, and thus $O(n^3 m)$ multiplications and additions in R.*

Proof. Write $S_r = \sum_{i=0}^{k-1} M^i$ and observe that $M^r = (M-1)S_r + 1$ and $S_{r+s} = M^r S_s + S_r$ for $r, s \in \mathbb{N} \setminus \{0\}$. This shows that at the beginning of Step (3), $T = S_{2^i}$ and $S = S_{k_i}$, where $k_i = \sum_{j=0}^{i} b_j 2^j$. □

Now we are ready to prove that complements of elementary abelian factors can be found using mere linear algebra.

7.6 Theorem. *Let $0 \le k \le n$ be such that G_{k+1} is an elementary abelian normal subgroup of G of exponent p. Then, using*

$$O(k(n-k)\log^3 p + k^4(n-k)^3 \log^2 p + k^2(n-k)^3(L_1 - L_{k+1})\log^2 p)$$

bit operations, it can be tested whether a complement H to G_{k+1} exists in G, and in this case, elements $u_1, \dots, u_k \in G_{k+1}$ can be computed such that $(g_1u_1, \dots, g_ku_k)$ is an induced generating sequence of H, and thus H is exhibited by the polycyclic generating sequence

$$(g_1u_1, \dots, g_ku_k, g_{k+1}, \dots, g_n).$$

Proof. Following [1, Section 3], G_{k+1} is complemented in G if, and only if, there exist $u_1, \dots, u_k \in G_{k+1}$ such that $h_1 = g_1u_1, \dots, h_k = g_ku_k$ satisfy a set of defining relations for G/G_{k+1} in the generators $G_{k+1}g_1, \dots, G_{k+1}g_k$. Such a set of defining relations can be easily derived from the polycyclic presentation of G, by replacing g_i by $G_{k+1}g_i$ throughout.

Now let $1 \le i < j \le k$ and write $g_jg_i = g_1^{a_1} \dots g_k^{a_k} u$ with $u \in G_{k+1}$ and reduced right hand side, then

$$h_jh_i = g_jg_iu_j^{g_i}u_i = g_1^{a_1} \dots g_k^{a_k} u u_j^{g_i} u_i. \tag{$*$}$$

Moreover, putting

$$u_i^* = u_1^{g_1^{a_1-1}} \dots u_1^{g_1} u_1,$$

we obtain

$$\begin{aligned} h_1^{a_1} \dots h_k^{a_k} &= (g_1u_1)^{a_1}(g_2u_2)^{a_2} \dots (g_ku_k)^{a_k} \\ &= g_1^{a_1}u_1^* g_2^{a_2} u_2^* \dots g_k^{a_k} u_k^* \qquad (**) \\ &= g_1^{a_1} g_2^{a_2} \dots g_k^{a_k} (u_1^*)^{g_2^{a_2} \dots g_k^{a_k}} (u_2^*)^{g_3^{a_3} \dots g_k^{a_k}} \dots (u_{k-1}^*)^{g_k^{a_k}} u_k^* \end{aligned}$$

Equating $(*)$ and $(**)$, the u_j have to satisfy

$$u u_j^{g_i} u_i = (u_1^*)^{g_2^{a_2} \dots g_k^{a_k}} (u_2^*)^{g_3^{a_3} \dots g_k^{a_k}} \dots (u_{k-1}^*)^{g_k^{a_k}} u_k^*. \qquad (***)$$

With respect to the basis $g_{k+1}, \dots, g_n$ of G_{k+1} (viewed as a $\mathbb{F}_p$-vector space), write $u_j = (x_{j,k+1}, \dots, x_{j,n})$. If $k < l \le n$, then $g_l^{g_i} = (e_{l,k+1}, \dots, e_{l,k+1})$ with respect to that basis, so that the action of g_i on u_l can be described by multiplying the vector by the matrix $E = (e_{l,m})$, which can be read off the presentation of G. Note that u_l^* corresponds to

$$(x_{l,k+1}, \dots, x_{l,n})(E^{a_1-1} + E^{a_1-2} + \dots + E + 1)$$

By Lemma 7.5, the matrix $E^{a_1-1} + E^{a_1-2} + \dots + E + 1$ can be computed using $O(\log a_i (n-k)^3 \log^2 p)$ bit operations. Thus, computing matrices corresponding to the action of $g_k^{a_k}$, $g_{k-1}^{a_{k-1}} g_k^{a_k}$, $\dots$, $g_2^{a_2} \dots g_k^{a_k}$ in that order, a linear expression of the coefficients of the right hand side of $(***)$ can be obtained using $O((L_1 - L_{k+1})(n-k)^3 \log^2 p)$ bit operations. The same is true for the left hand side. Comparing coefficients leads to $n-k$ linear equations in $x_{l,m}$, $1 \le l \le k < m \le n$.

Similarly, if $1 \le i \le k$, write $g_i^{p_i} = g_1^{a_1} \dots g_k^{a_k} u$, where the right hand side is reduced with $u \in G_{k+1}$. Then $h_i^{p_i} = g_i^{p_i} u^{**} u$, where $u^{**} = u_i^{p_i - 1} u_i^{p_i - 2} \dots u_i$. This leads to

$$u^{**} u = (u_1^*)^{g_2^{a_2} \dots g_k^{a_k}} (u_2^*)^{g_3^{a_3} \dots g_k^{a_k}} \dots (u_{k-1}^*)^{g_k^{a_k}} u_k^*,$$

and thus to $n-k$ linear equations, at a cost of $O((n-k)^2 (L_1 - L_{k+1}) \log^2 p)$ bit operations.

Thus, at a total cost of $O(k^2 (L_1 - L_{k+1})(n-k)^3 \log^2 p)$ bit operations, we obtain a system of $\frac{1}{2}k(k+1)(n-k)$ equations in $k(n-k)$ variables. Gaussian elimination requires at most $O(k(n-k) \log^3 p + k^4 (n-k)^3 \log^2 p)$ bit operations to solve the system, or to deduce that none exists. □

7.7 Remark. As in [1, Section 3], the algorithm outlined in the proof of Theorem 7.6 can also be used to compute the set of all complements, or of one representative of each conjugacy class complements of G_{k+1} in G. The cost for computing an induced generating sequence for each additional complement is polynomial in k, $n-k$ and $\log p$; however, trivial examples show that the number of complements or representatives need not.

8. Reduction to the nilpotent case

In order to prove the main result of this section, we will need some technical results.

8.1 Lemma. *Assume that G_{k+1} is an elementary abelian normal subgroup of G of exponent p, and that $N_1 < N_2 < \dots < N_r \le G_{k+1}$ are subgroups of G given by polycyclic generating sequences.*

(a) *Using*

$$O((n-k)\log^3 p + k(n-k)^3\log^2 p + k^2(n-k)^2\log^2 p)$$

bit operations, a polycyclic presentation of G with generators $g_1,\ldots,g_k,h_{k+1},\ldots,h_n$ can be computed such that $G_{k+1} = \langle h_{k+1},\ldots,h_n\rangle$, $U = \langle h_{r+1},\ldots,h_n\rangle$ and $V = \langle h_{s+1},\ldots,h_n\rangle$.

(b) *Converting r reduced words in $g_1,\ldots,g_k,h_{k+1},\ldots,h_n$ into reduced words in $g_1,\ldots,g_n$ requires $O(r(n-k)^2\log^2 p)$ bit operations; for the opposite direction, $O((n-k)\log^3 p + (n-k)^3\log^2 p + r(n-k)^2\log^2 p)$ bit operations are required.*

Proof. We consider G_{k+1} as a vector space with basis $g_{k+1},\ldots,g_n$ over $\mathbb{F}_p$, so that polycyclic generating sequences of its subgroups correspond to bases of subspaces. Now extend a basis of N_1 to a basis of N_2, then on to N_3, and so forth, and finally to one of G_{k+1}, and let $h_{k+1},\ldots,h_n$ be the corresponding polycyclic generating system of G_{k+1}. Computing such a basis requires $n-k$ inverses and $(n-k)^3$ multiplications modulo p, and thus $O((n-k)\log^3 p + (n-k)^3\log^2 p)$ bit operations.

Now $(g_1,\ldots,g_k,h_{k+1},\ldots,h_n)$ is a polycyclic generating sequence of G, and any reduced word in $g_1,\ldots,g_k,h_{k+1},\ldots,h_n$ can be rewritten as a word in $g_1,\ldots,g_n$ at the cost of a matrix multiplication, requiring $O((n-k)^2\log^2 p)$ bit operations. To convert reduced words in $g_1,\ldots,g_k,h_{k+1},\ldots,h_n$, the inverse matrix has to be computed, at a cost of $O((n-k)\log^3 p + (n-k)^3\log^2 p)$ bit operations. Rewriting each word then requires $O((n-k)^2\log^2 p)$ bit operations. This proves (b).

In order to compute a polycyclic presentation of G with generators

$$g_1,\ldots,g_k,h_{k+1},\ldots,h_n,$$

proceed as follows. For relations $g_i^{e_i} = w_{i,i}$ and $g_j^{g_i} = w_{i,j}$, rewrite the $w_{i,j}$ as words in $g_1,\ldots,g_k,h_{k+1},\ldots,h_n$ to obtain a right hand side in the new generators. Right hand sides of relations whose left hand side is of the form $h_j^{g_i}$ can be found by conjugating the matrix action of g_i on $g_{k+1},\ldots,g_n$ by the matrix affording the base change from $g_{k+1},\ldots,g_n$ to $h_{k+1},\ldots,h_n$. Note that $h_j^p = 1 = h_j^{h_i}$ for all i, j with $k < i, j \leq n$. Thus, the total cost of computing a polycyclic presentation with generators $g_1,\ldots,g_k,h_{k+1},\ldots,h_n$ is $O((n-k)\log^3 p + k(n-k)^3\log^2 p + k^2(n-k)^2\log^2 p)$. □

Next, we extend Theorem 7.6 to arbitrary elementary abelian factors.

8.2 Proposition. *Let $0 \leq k < m \leq n$ be integers such that G_{k+1} and G_{m+1} are normal G such that G_{k+1}/G_{m+1} elementary abelian of prime exponent p. Then there exists an algorithm which, at a cost of*

$$O(((m-k)^6 + k^4(m-k)^3 + k^2(m-k)^3(L_1 - L_{k+1}))(I_{k+1} - I_{m+1}))$$

bit operation, either computes integers l, r with $k \leq l < r \leq m$, and a polycyclic generating sequence

$$(g_1u_1,\ldots,g_ku_k,u_{k+1},\ldots,u_m,g_{m+1},\ldots,g_n)$$

of G such that

$$(g_1u_1, \dots, g_ku_k, u_{k+1}, \dots, u_l, u_{r+1}, \dots, u_m, g_{m+1}, \dots, g_n)$$

is a polycyclic generating sequence of a maximal subgroup H of G and U/V is a chief factor of G avoided by H, where $U = \langle\, u_{l+1}, \dots, u_n \,\rangle$ and $V = \langle\, u_{r+1}, \dots, u_n \,\rangle$, or proves that $G_{k+1}/G_{m+1} \le \Phi(G/G_{m+1})$.

Proof. Note that a polycyclic presentation of G/G_{m+1} can be read off $\mathscr{P}$ (simply by deleting $g_{m+1}, \dots, g_n$ from the set of generators and each relation), and a polycyclic generating sequence of G/G_{m+1} with the above property can be translated into one of G exhibiting H by appending $g_{m+1}, \dots, g_n$. Thus, we may assume that $m = n$.

We consider G_{k+1} as an $\mathbb{F}_pG$-module. Repeated application of the Meat-Axe algorithm [7, 8] yields an $\mathbb{F}_pG$-composition series of the $\mathbb{F}_pG$-module G_{k+1}, at an expected cost of

$$\begin{aligned}&O((m-k)^7 \log^2 p + (m-k)^4 \log(m-k) \log^3 p + k(m-k)^4 \log^2 p)\\ \subseteq\ &O((m-k)^6 (L_{k+1} - L_{m+1}))\end{aligned}$$

bit operations. (See Remark 8.3 below.) This yields a polycyclic generating sequence $(u_{k+1}, \dots, u_n)$ of G_{k+1} refining a G-composition series

$$1 = N_0 \triangleleft N_1 \triangleleft \cdots N_r = G_{k+1}$$

of G_{k+1}. Using Lemma 8.1, at a cost of

$$O((m-k) \log^3 p + k(m-k)^3 \log^2 p + k^2(m-k)^2 \log^2 p)$$

bit operations, we compute a presentation with generators $g_1, \dots, g_k, x_{k+1}, \dots, x_n$.

For $i = 1, \dots, r$, we use Theorem 7.6 to test whether N_i/N_{i-1} has a complement H/N_{i+1} in G_{k+1}/N_{i-1}, at a total cost of

$$O(k(m-k)^2 \log^3 p + k^4(m-k)^4 \log^2 p + k^2(m-k)^4 (L_1 - L_{k+1}) \log^2 p).$$

If no such complement exists, then by [2, A, 9.10], we have $N_i/N_{i-1} \le \Phi(G/N_{i-1})$ for all i. Therefore, by [2, A, 9.2 (e)], G_{k+1} is contained in the Frattini subgroup of G, and the proof is complete.

Therefore, let i be minimal such that N_i/N_{i-1} is complemented in G/N_{i+1}, and let l, r be such that $N_i = \langle\, x_{l+1}, \dots, x_n \,\rangle$ and $N_{i+1} = \langle\, x_{r+1}, \dots, x_n \,\rangle$. By Theorem 7.6, there exist reduced words $v_1, \dots, v_k$ in the generators $x_{k+1}, \dots, x_n$ such that

$$(g_1v_1, \dots, g_kv_k, x_{k+1}, \dots, x_n)$$

is a polycyclic generating sequence of G such that

$$(g_1v_1, \dots, g_kv_k, x_{k+1}, \dots, x_l, x_{r+1}, \dots, x_n)$$

is a polycyclic generating sequence of H.

Finally, $v_1, \ldots, v_k$ can be rewritten as reduced words $u_1, \ldots, u_k$ in $g_{k+1}, \ldots, g_n$ by substituting reduced expressions of $u_{k+1}, \ldots, u_n$, at a cost of $O(k(n-k)^2 \log^2 p)$ bit operations. Using $(m-k)\log^3 p \in O(L_{k+1} - L_{m+1})$, we obtain the bound on the cost stated in the proposition. □

8.3 Remark. The only probabilistic part of the above algorithm is the use of the Meat-Axe algorithm. Selecting evenly distributed random elements in the relevant matrix algebra seems to require the computation of a basis of that algebra, at a cost of $O((m-k)^6 \log^2 p + (m-k)^2 \log^3 p)$ bit operations. The remainder of the algorithm consists of computing the characteristic polynomial f of such a random element A, at a cost of $O((m-k)^3 \log^2 p + (m-k)\log^3 p)$, factoring that polynomial, at an expected cost of $O((m-k)^3 \log(m-k) \log^3 p)$ [4], finding a nontrivial vector in the nullspace of $g(A)$ using Gaussian elimination, where g is an irreducible factor of f, at a cost of $O((m-k)\log^3 p + (m-k)^4 \log^2 p)$, and computing submodules generated by a single vector, at the cost of $k(m-k)^3 \log^2 p + (m-k)\log^3 p)$. This gives an expected running time of

$$O((m-k)^6 \log^2 p + (m-k)^3 \log(m-k) \log^3 p + k(m-k)^3 \log^2 p).$$

Note that r elements sufficient for practical purposes can be obtained by multiplying and adding a certain number of randomly selected generators, at the cost of $O((m-k)^3 \log^2 p)$, essentially reducing the above estimate to

$$O((m-k)\log^3 p + (m-k)^4 \log^2 p).$$

In order to apply Proposition 8.2 to an exhibited subgroup, the next lemma will prove useful.

8.4 Lemma. *Let $0 \le k \le m \le n$ be such that G_{k+1} and G_{m+1} are normal subgroups of G, and such that G_{k+1}/G_{m+1} is elementary abelian of exponent p. Assume that H is a subgroup exhibited by $(g_1, \ldots, g_n)$, and write $J = \{\, i \mid g_i \in H \,\}$. Let $h_1, \ldots, h_n \in G$ be such that $h_i = g_i$ if $i \notin J$ or $i > m$, such that $h_i = g_i x_i$ if $i \in J$ and $i \le k$, where $x_i \in H \cap G_{k+1}$, and such that $G_{k+1} \cap H = \langle\, h_j \mid j \in J, j > k \,\rangle$. Then $(h_1, \ldots, h_n)$ is a polycyclic generating sequence of G. More generally, if $E \ge H \cap G_{k+1}$ is exhibited by $\mathscr{P}$ and $J = \{\, i \mid g_i \in E \,\}$, then $(h_i)_{i \in J}$ is a polycyclic generating sequence of E.*

Proof. We first show that $(h_1, \ldots, h_n)$ is a polycyclic generating sequence of G. Put $H_i = \langle\, h_i, \ldots, h_n \,\rangle$. As G_{k+1}/G_{m+1} is a $\mathbb{F}_p$-vector space with basis

$$\{\, G_{m+1} g_j \mid k < j \le m \,\},$$

the factor space $G_{k+1}/(G_{k+1} \cap H)$ has a basis

$$\{\, (G_{k+1} \cap H) g_i \mid k < i \le m, i \notin J \,\} = \{\, (G_{k+1} \cap H) h_i \mid k < i \le m, i \notin J \,\}.$$

Moreover,

$$\{\, G_{m+1} g_i \mid k < i \le m, i \in J \,\}$$

is a basis of $(G_{k+1} \cap H)/G_{m+1}$, and since

$$\{ G_{m+1}h_i \mid k < i \leq m, i \in J \}$$

generates $(G_{k+1} \cap H)/G_{m+1}$, it follows that it is likewise a basis. As $H_i = G_i$ if $i \leq k$ or $i > m$, this shows that

$$G = H_1 \triangleright H_2 \triangleright \cdots \triangleright H_n \triangleright H_{n+1} = 1$$

is a series with cyclic factors, with $|H_i : H_{i+1}| = |G_i : G_{i+1}|$ for all i, and $(h_1, \ldots, h_n)$ is a polycyclic generating sequence.

Since E contains $H \cap G_{k+1}$, we have $g_i \in E$ if, and only if, $h_i \in E$ for all i. Therefore E is also exhibited by $(h_1, \ldots, h_n)$. □

We are now ready to prove the main result of this section: the reduction of a multiplication in a finite soluble group to multiplications in nilpotent subgroups. To formulate the result, let $\mathbf{P}_{\text{nilp}}$ and $\mathbf{M}_{\text{nilp}}$ be functions such that r multiplications in an arbitrary nilpotent subgroup N of G given by a polycyclic presentation refining an elementary abelian normal series of N can be carried out using $\mathbf{P}_{\text{nilp}}(G) + r\mathbf{M}_{\text{nilp}}(G)$ bit operations.

8.5 Theorem. *Assume that $\mathscr{P}$ refines a normal series with elementary abelian factors. Then there exists a Las Vegas algorithm which, at an expected cost of $O(n\mathbf{P}_{\text{nilp}} + n^6L^4\mathbf{M}_{\text{nilp}} + n^9I + n^7LI)$ bit operations, computes a polycyclic presentation $\mathscr{P}'$ of G such that r multiplications with respect to $\mathscr{P}'$ require at most $O(n\mathbf{P}_{\text{nilp}} + rn^2L^2\mathbf{M}_{\text{nilp}}(G))$ bit operations. Moreover, r normal words in the generators of $\mathscr{P}'$ can be written as a normal word with respect to $\mathscr{P}$ at a cost of $O(n\mathbf{P}_{\text{nilp}} + rnL\mathbf{M}_{\text{nilp}} + nI)$ bit operations, and a normal word in the generators of $\mathscr{P}$ can be written as a normal word with respect to $\mathscr{P}'$ at a cost of $O(n\mathbf{P}_{\text{nilp}} + rL\mathbf{M}_{\text{nilp}})$ bit operations. Therefore, we may choose $\mathbf{P}_{\text{elab}} = n\mathbf{P}_{\text{nilp}} + n^6L^4\mathbf{M}_{\text{nilp}} + n^9I + n^7LI$ and $\mathbf{M}_{\text{elab}} = nL\mathbf{M}_{\text{nilp}}$.*

Proof. Denote the terms of an elementary abelian series of G refined by $\mathscr{P}$ by

$$1 = L_0 \triangleleft L_1 \triangleleft \cdots \triangleleft L_t = G.$$

We construct polycyclic presentations $\mathscr{P}_0, \mathscr{P}_1, \ldots, \mathscr{P}_n$ as follows. Let $\mathscr{P}_0 = \mathscr{P}$, $H_0 = G$ and $N_0 = 1$. Now assume that $\mathscr{P}_i$ has generators $h_1, \ldots, h_n$ exhibiting subgroups H_i, $N_1, \ldots, N_i$ such that for $j = 1, \ldots, i$, N_j is normalised by $N_{j+1}, \ldots, N_i$ and H_i, all N_j are nilpotent, $H_i \cap N_j$ is a term of the polycyclic series defined by $\mathscr{P}_i$, and $G = H_iN_iN_{i-1}\ldots N_1$. Moreover, assume that reduced expressions of $g_1, \ldots, g_n$ in the generators of $\mathscr{P}_i$ are known.

If $H_i = 1$, let $\mathscr{P}_{i+1} = \mathscr{P}_i$ and $H_{i+1} = N_{i+1} = 1$. Otherwise, restrict $\mathscr{P}_i$ to H_i to obtain a polycyclic presentation $\overline{\mathscr{P}_i}$ of H_i with generators $x_1, \ldots, x_m$. This restriction refines the series with terms $H_i \cap N_j$, $H_i \cap L_k$, where $1 \leq j \leq i$ and $1 \leq k \leq t$. Denote this series by

$$1 = M_0 \triangleleft M_1 \triangleleft \cdots \triangleleft M_s = H_i.$$

Starting with $k = 1$, use Proposition 8.2 to test whether $M_k/M_{k-1} \leq \Phi(H_i/M_{k-1})$. If this is the case, then also $M_k \leq \Phi(H_i)$, see, e. g. [2, A, 9.2 (e)]. Thus, we may replace k by $k - 1$ until we obtain a polycyclic generating sequence $y_1, \dots, y_m$ of H_i likewise refining the series with terms $H_i \cap K_j$, $H_i \cap L_k$, where $1 \leq j \leq i$ and $1 \leq k \leq t$. Moreover, Proposition 8.2 yields a term $N_{i+1} \leq M_k$ of the polycyclic series defined by $(y_1, \dots, y_m)$ and a subgroup H_{i+1} of H_i exhibited by $(y_1, \dots, y_m)$ such that $H_{i+1}M_k = H_i$. The expected total cost is $O(n^8 I + n^6 LI)$ bit operations. Since since H_i normalises $N_1, \dots, N_i$, the same is true for H_{i+1} and N_{i+1}. By construction, H_{i+1} also normalises N_{i+1}. Let l be such that $L_{l-1} \cap H_i \leq M_{k-1} < M_k \leq L_l$. As L_l/L_{l-1} is (elementary) abelian and $M_{k-1} \leq \Phi(H_i)$, it follows from [2, A, 9.3 (c)] that $K_{i+1} = L_l \cap H_i$ and N_{i+1} are nilpotent. For $j = i, \dots, 1$, let $K_j = K_{j+1}N_j \trianglelefteq H_j$, and observe that K_{j+1} normalises N_j, and that the N_j are terms in the polycyclic sequence defined by the restriction of $\mathscr{P}_i$ to H_j. By Proposition 7.1, a multiplication in K_j can be carried out using one multiplication in K_{j+1} and $O(nL^2)$ multiplications in N_j. Since $L_l \leq K_1$, it follows that a multiplication in L_l requires $O(n\mathbf{P}_{\text{nilp}} + n^2L^2\mathbf{M}_{\text{nilp}})$ bit operations.

We now define a sequence $(h'_1, \dots, h'_n)$ of G by replacing the x_i among the generators of $\mathscr{P}_i$ by the corresponding y_i, and by keeping the others. By Lemma 8.4, $(h'_1, \dots, h'_n)$ is a polycyclic generating sequence of G.

Using Proposition 4.6, the associated polycyclic presentation $\mathscr{P}_{i+1}$ can be computed at a cost of $O(n^3L^2)$ multiplications in L_l and $O(nI)$ bit operations. We obtain reduced expressions of $g_1, \dots, g_n$ in $(h'_1, \dots, h'_n)$ at a cost of $O(n^2L^2)$ multiplications in L_l and $O(n^2I)$ bit operations. By Proposition 8.2, $N_{i+1}/N_{i+1} \cap H_{i+1}$ is a chief factor of H_i. Therefore if $j \leq i$, either $N_j \cap H_i \leq N_{i+1} \cap H_{i+1}$, or $N_{i+1} \leq N_j$. In the first case, the $h_i \in N_j$ are the same as the h'_i in N_j, and therefore N_j is exhibited by $(h'_1, \dots, h'_n)$. In the second case, the same result follows from Lemma 8.4. This shows that $\mathscr{P}_{i+1}$ has the required properties.

Finally, by the above argument, a multiplication in $\mathscr{P}' = \mathscr{P}_n$ can be carried out using $O(n\mathbf{P}_{\text{nilp}} + (n^2L^2\mathbf{M}_{\text{nilp}})$ bit operations, while the computation of $\mathscr{P}'$ requires

$$O(n\mathbf{P}_{\text{nilp}} + n^9 I + n^7 LI + n^6 L^4 \mathbf{M}_{\text{nilp}})$$

bit operations. □

9. Reduction to *p*-groups and conclusions

Finally, we show that the problem of computing in a nilpotent group G can be reduced to computing in p-groups, where p ranges over all primes dividing the order of G.

Let $\mathscr{S}$ be a set of Sylow subgroups of G containing exactly one p-Sylow subgroup for each prime dividing the order of G. We define

$$\mathbf{P}_p(G) = \sum_{P \in \mathscr{S}} \mathbf{P}_{\text{elab}}(P) \qquad \text{and} \qquad \mathbf{M}_p(G) = \sum_{P \in \mathscr{S}} \mathbf{M}_{\text{elab}}(P).$$

9.1 Theorem. *Let G be a nilpotent group. Then a polycyclic presentation $\mathscr{P}'$ of G having the same associated polycyclic series as $\mathscr{P}$ and exhibiting each Sylow subgroup of G can be computed using*

$$O(n\mathbf{P}_p(G) + n^2 I + LI + n^4 L^2 \mathbf{M}_p(G))$$

bit operations. Moreover, a reduced word in $g_1, \ldots, g_n$ can be translated into a reduced word with respect to $\mathscr{P}'$ using $O(L)$ multiplications with respect to $\mathscr{P}'$, and a reduced word with respect to $\mathscr{P}'$ can be translated into a reduced word in $g_1, \ldots, g_n$ using $O(nL)$ multiplications with respect to $\mathscr{P}'$. Thus, we may choose $\mathbf{P}_{\text{nilp}} = n\mathbf{P}_p(G) + n^2 I + LI + n^4 L^2 \mathbf{M}_p(G)$ and $\mathbf{M}_{\text{nilp}}(G) = nL\mathbf{M}_p(G)$.

Proof. Let $i \geq 1$ be such that the Sylow subgroups of G_{i+1} are exhibited by $(g_{i+1}, \ldots, g_n)$, and let $p = p_i$.

By hypothesis, the order of g_j is a power of p_j if $i < j \leq n$, and elements of G_{i+1} of coprime order commute. Therefore, at a cost of $O((n-i)L_{i+1})$ bit operations per relation, or a total of $O((n-i)^2 L_{i+1})$ bit operations, we may temporarily re-order $g_{i+1}, \ldots, , g_n$ such that $p_j \neq p$ if $i < j \leq m$ and $p_j = p$ if $m < j \leq n$, for a suitable integer m. Thus, G_{m+1} is a p-Sylow subgroup of G_{i+1}.

Let $g = g_i$, $w = w_{i,i}(= g^p)$, $j = i + 1$.

As in the proof of Theorem 7.6, the relation $g^p = w$ suffices to compute an element $u \in G_j$ such that $H/G_{j+1} = \langle G_{j+1}, gu \rangle / G_{j+1}$ is a complement of G_j/G_{j+1} in $\langle g, G_j \rangle / G_{j+1}$; this requires $O(l_j^3 + l_i l_j^2)$ bit operations. If $j < m$, we replace j by $j+1$, g by gu and w by the normal form of $u^{g^{p-1}} u^{g^{p-2}} \ldots uw$. Note that the latter can be computed at a cost of $O(nl_i L_{j+1})$ multiplications in L_{j+1} by Lemma 3.2. Thus, we arrive at an induced generating sequence $g_1, \ldots, g_{i-1}, g_i v, g_{i+1}, \ldots, g_n$ with $v \in G_{i+1}$, such that $(g_i v)^p \in G_{m+1}$. Therefore $\langle G_{m+1}, g_i v \rangle$ is a Sylow p-subgroup of G_i. By Proposition 4.6, a polycyclic presentation with generators $g_1, \ldots, g_{i-1}, g_i v, g_{i+1}, \ldots, g_n$ can be computed using $O((n-i)^3 L_{i+1}^2)$ multiplications in G_{i+1} and $O((n-i)I_{i+1})$ bit operations.

Since the Sylow q-subgroups of G_{i+1} coincide with those of G_i if $q \neq p$, we have obtained a polycyclic presentation of G exhibiting all Sylow subgroups of G_i, at a cost of $O((n + l_i)I_{i+1})$ bit operations and $O(n^3 L_{i+1}^2)$ multiplications L_{i+1}. By hypothesis, each of these can be carried out using $\sum_{p \mid |G|} f(|G|_p))$ bit operations, multiplying subwords in Sylow subgroups. Using the fact that there are only n possible values for i, we the first result follows. Note that a reduced word with respect to $\mathscr{P}'$ can be translated into a reduced word with respect to $\mathscr{P}$ and back at a cost of $O(nL)$ multiplications and $O(L)$ multiplications with respect to $\mathscr{P}'$, respectively, by Proposition 4.5. □

9.2 Remark. Note that the algorithm outlined in the proof of Theorem 9.1 does not change the polycyclic series associated to $\mathscr{P}$. Therefore, in the proof of Theorem 8.5, it is possible to replace the restriction of $\mathscr{P}_i$ to $L_l \cap H_i$ by the polycyclic generating sequence produced by Theorem 9.1, at no further cost. Let π be a set of primes. Then the restriction of $\mathscr{P}'$ to any N_i exhibits the (unique) Hall π-subgroup $(N_i)_\pi$ of N_i. As a consequence, multiplications in the N_i reduce to computations in their Sylow subgroups. Moreover, since N_i normalises N_j if $i > j$, then also $(N_i)_\pi$ normalises $(N_j)_\pi$, so that $(N_n)_\pi(N_{n-1})_\pi \dots (N_1)_\pi$ is a Hall π-subgroup of G exhibited by $\mathscr{P}'$. It is now easy to see that these Hall subgroups form a Hall system of G. (For a definition of Hall systems, see, e. g. [2, I, 4.1].)

The results of Sections 6 – 9 can now be summarised as follows.

9.3 Corollary. *Let $\mathscr{P}$ be an arbitrary polycyclic presentation, and assume that the prime factorisation of each e_i is known. Then r multiplications with respect to p can be carried out using an expected number of*

$$O(n^4\mathbf{P}_p(G) + l^9L^5\mathbf{M}_p(G) + rl^3L^3\mathbf{M}_p(G))$$

bit operations, where l is the composition length of G.

Proof. This follows by substituting the functions for $\mathbf{P}$, $\mathbf{M}$, $\mathbf{P}_{\mathrm{comp}}$, $\mathbf{M}_{\mathrm{comp}}$, $\mathbf{P}_{\mathrm{elab}}$, $\mathbf{M}_{\mathrm{elab}}$, $\mathbf{P}_{\mathrm{nilp}}$, and $\mathbf{M}_{\mathrm{nilp}}$ given in Proposition 6.1, Theorem 6.2, Theorem 8.5, and Theorem 9.1 into each other and using the obvious inequalities $n \leq l \leq L$, $I \leq L^3$. $\square$

References

1. Celler, F., Neubüser, J., and Wright, C. R. B.: Some remarks on the computation of complements and normalizers in soluble groups. *Acta Appl. Math.* **21** (1990), 57–76.
2. Doerk, K. and Hawkes, T.: *Finite soluble groups*. de Gruyter, Berlin, 1992.
3. The GAP Group: *GAP — Groups, Algorithms, and Programming, Version 4.3*. St Andrews, Aachen, 2002, `http://www-gap.dcs.st-and.ac.uk/~gap`.
4. von zur Gathen, J. and Gerhard, J.: *Modern Computer Algebra*. Cambridge University Press, Cambridge, 1999.
5. Gebhardt, V.: Efficient collection in infinite polycyclic groups. To appear in *J. Symbolic Comp.*
6. Glasby, S. P.: The composition and derived lengths of a soluble group. *J. Algebra* **120** (1989), 406–413.
7. Holt, D. F. and Rees, S.: Testing modules for irreducibility. *J. Austral. Math. Soc. Ser. A* **57** (1994), 1–16.
8. Ivanyos, G. and Lux, K.: Treating the exceptional cases of the MeatAxe. *Experiment. Math.* **9** (2000), 373–381.
9. Leedham-Green, C. R. and Soicher, L.: Collection from the left and other strategies. *J. Symbolic Comp.* **9** (1990), 665–675.
10. Leedham-Green, C. R. and Soicher, L.: Symbolic collection using Deep Thought. *LMS J. Comput. Math.* **1** (1998), 9–24.
11. Sims, C. C.: *Computation with Finitely Presented Groups,*. Cambridge University Press, 1994.

12. Vaughan-Lee, M. R.: Collection from the Left. *Journal of Symbolic Comp.* **9** (1990), 725–733.