Journal of Algebra 352 (2012) 430–450

Contents lists available at SciVerse ScienceDirect

Journal of Algebra

www.elsevier.com/locate/jalgebra

An effective version of the Lazard correspondence

Serena Cicalò [a,1], Willem A. de Graaf [b,*], Michael Vaughan-Lee [c,2]

[a] *Dipartimento di Matematica e Informatica, Università di Cagliari, Italy*
[b] *Dipartimento di Matematica, Università di Trento, Italy*
[c] *Christ Church, Oxford, United Kingdom*

ARTICLE INFO

Article history:
Received 6 May 2011
Available online 22 December 2011
Communicated by Eamonn O'Brien

Keywords:
p-Groups
Lie rings
Lazard correspondence
Effective methods

ABSTRACT

The Lazard correspondence establishes an equivalence of categories between p-groups of nilpotency class less than p and nilpotent Lie rings of the same class and order. The main tools used to achieve this are the Baker–Campbell–Hausdorff formula and its inverse formulae. Here we describe methods to compute the inverse Baker–Campbell–Hausdorff formulae. Using these we get an algorithm to compute the Lie ring structure of a p-group of class $< p$. Furthermore, the Baker–Campbell–Hausdorff formula yields an algorithm to construct a p-group from a nilpotent Lie ring of order p^n and class less than p. At the end of the paper we discuss some applications of, and practical experiences with, the algorithms.

1. Introduction

It has been known since the 1950s that the Baker–Campbell–Hausdorff formula gives an isomorphism between the category of nilpotent Lie rings with order p^n and nilpotency class c and the category of finite p-groups with order p^n and nilpotency class c, provided $p > c$. This is known as the Lazard correspondence [5, p. 91]. It also gives an isomorphism between the category of nilpotent Lie algebras over the rationals $\mathbb{Q}$ and the category of torsion free radicable nilpotent groups. This is known as the Mal'cev correspondence [18]. For an in-depth account of these matters we refer to [15, Chapters 9 and 10].

* Corresponding author.
E-mail addresses: cicalo@science.unitn.it (S. Cicalò), degraaf@science.unitn.it (W.A. de Graaf), michael.vaughan-lee@chch.ox.ac.uk (M. Vaughan-Lee).

[1] Serena Cicalò thanks the Department of Mathematics of the University of Trento for its hospitality in 2011. She gratefully acknowledges the support of the programme "Master & Back", PR-MAB-A2009-837.
[2] Michael Vaughan-Lee thanks the CIRM Trento for its hospitality during four weeks in January/February 2011.

0021-8693/$ – see front matter
doi:10.1016/j.jalgebra.2011.11.031

Using the Lazard correspondence we can transform questions on p-groups (of class $< p$) to questions on Lie rings. This was exploited in the classification of groups of order p^6 and p^7 for $p > 5$ (see [20,23]). Underlying the group classifications are classifications of the nilpotent Lie rings of order p^6 and p^7. The Baker–Campbell–Hausdorff formula was used to turn Lie ring presentations into group presentations.

A related application of the Lazard correspondence was given by Evseev in [9], where he proves Higman's PORC conjecture for a certain class of p-groups, by translating it to a question on Lie rings, and then solving it in that setting.

The Mal'cev correspondence was exploited in [2] to provide a fast method of multiplication in infinite polycyclic groups.

In the next section we will describe the Baker–Campbell–Hausdorff formula (or BCH-formula for short).

In this paper we describe computational methods to perform the Lazard correspondence in practice. That is, methods to construct the Lie ring corresponding to a given p-group (of class $< p$), and vice versa, to find the p-group corresponding to a nilpotent Lie ring of order p^n and class $< p$. For this we need to explicitly compute various formulae. First there is the BCH-formula itself, for which we use a method from [21], which we briefly recall in Section 3. Secondly we need the BCH inverse formulae. The problem of computing these does not seem to have been considered in the literature before. Methods for computing the inverse formulae are detailed in Sections 4 to 6. In Section 4 we define some of the notation that we use. Section 5 is devoted to a method for computing the homogeneous components of repeated commutators, which can be viewed as BCH-formulae of higher order. Then in Section 6 it is shown how to use this to compute the inverse BCH formulae. In Section 7 we then describe how to compute the Lie ring structure of a given p-group of class $< p$.

The method can also be used to compute a Lie algebra corresponding to a T-group (see Section 7 for the definition), which is the Lie algebra of the *radicable hull* of the group under the Mal'cev correspondence. Algorithms for this task are also given in [2]. One of these is based on the BCH-formula, and is similar to, but different from, ours. The implementation of this algorithm (in the GAP package Guarana [1]) is able to deal with T-groups up to class 9.

In Section 8 we report on practical experiences with our implementation of the algorithms in MAGMA [4]. Currently our programs are able to deal with p-groups and Lie rings of class up to 14. Experiments with the algorithms for computing the various formulae (see Section 8) suggest that this might be extended to class 15 or 16. However, it seems unlikely that we will be able to go beyond that.

Finally, in the last section we discuss two applications: one is to computing Hall-polynomials, which give an algorithm for computing products of elements in a p-group (or in a T-group), the second is to computing faithful representations of nilpotent Lie algebras over $\mathbb{Q}$, using the Mal'cev correspondence.

The first two authors have written a GAP package, LieRing [8], that contains, among other things, an implementation of the algorithms outlined in this paper for computing the Lazard correspondence.

In [11] Glauberman extended the construction of Lazard to certain p-groups of class $\geqslant p$. However, under this construction, it is not clear to which extent the group may be recovered from the Lie ring (cf. [11, Remark 6.11]). It would be interesting to see whether our methods can be extended to cover also Glauberman's construction, and then to investigate this question experimentally.

Throughout the paper, commutators play an important role. We will use the bracket notation $[\,,\,]$ both for the commutator in a group ($[g,h] = g^{-1}h^{-1}gh$) and in a Lie ring. From the context it will be clear which we mean. Sometimes we add an index (i.e., $[\,,\,]_G$ or $[\,,\,]_L$) for greater clarity. Also, if a, b are elements of an associative algebra, then $[a,b] = ab - ba$. For commutators of weight greater than two we use the right-normed convention. Thus

$$[x,x,y] = \big[x,[x,y]\big] \quad \text{and} \quad [x,y,x,y] = \big[x,\big[y,[x,y]\big]\big],$$

and so on.

2. The Baker–Campbell–Hausdorff formula

Let A be the free associative algebra with unity over the rationals $\mathbb{Q}$ which is freely generated by non-commuting indeterminates x, y. We have that A is spanned by the words in x and y. The *weight* of such a word is defined to be the number of occurrences of x and y. We extend A to the ring $\widehat{A}$ of formal power series consisting of the formal sums

$$\sum_{n\geqslant 0} u_n,$$

where u_n is a homogeneous element of weight n in A. If $a \in \widehat{A}$, and if the homogeneous component of a of weight 0 is 0, then we define

$$e^a = 1 + a + \frac{a^2}{2!} + \frac{a^3}{3!} + \cdots,$$

in the usual way. The product $e^x e^y \in \widehat{A}$ can be expressed in the form $1+u$ for some $u \in \widehat{A}$ with 0 as its homogeneous component of weight 0, and

$$e^x e^y = e^{z(x,y)},$$

where

$$z(x, y) = \sum_{n\geqslant 1} (-1)^{n-1} \frac{u^n}{n}.$$

The Baker–Campbell–Hausdorff formula (see, for example, [14, §V.5]) gives z as a linear combination of commutators in x and y, with rational coefficients. The first few components are given by

$$\begin{aligned} z(x, y) = x + y + \frac{1}{2}[x, y] + \frac{1}{12}[x, x, y] - \frac{1}{12}[y, x, y] - \frac{1}{24}[y, x, x, y] - \frac{1}{720}[x, x, x, x, y] \\ - \frac{1}{120}[x, y, x, x, y] - \frac{1}{360}[x, y, y, x, y] + \frac{1}{360}[y, x, x, x, y] + \frac{1}{120}[y, y, x, x, y] \\ + \frac{1}{720}[y, y, y, x, y] + \frac{1}{1440}[y, x, x, x, x, y] + \cdots. \end{aligned}$$

It turns out that all the homogeneous components of z are Lie elements of A (that is, elements in the Lie subalgebra of A generated by x and y with respect to the Lie product $[a, b] = ab - ba$), cf. [29, Theorem 2.5.4].

A similar formula holds for commutators:

$$\left[e^x, e^y\right] = e^{w(x,y)},$$

where

$$\begin{aligned} w(x, y) = [x, y] - \frac{1}{2}[x, x, y] - \frac{1}{2}[y, x, y] + \frac{1}{6}[x, x, x, y] + \frac{1}{4}[y, x, x, y] + \frac{1}{6}[y, y, x, y] \\ - \frac{1}{24}[x, x, x, x, y] - \frac{1}{12}[x, y, y, x, y] - \frac{1}{12}[y, x, x, x, y] - \frac{1}{24}[y, y, y, x, y] \\ + \frac{1}{120}[x, x, x, x, x, y] + \cdots. \end{aligned}$$

(Here $[e^x, e^y]$ is the group commutator $e^{-x}e^{-y}e^xe^y$, and $w(x, y)$ is an infinite sum of Lie elements in A.)

Under some circumstances we can use these formulae to turn a Lie algebra (or a Lie ring) into a group: if a and b are two elements in a Lie algebra L, define a group product $*$ on L and the group commutator $[\,,]_G$ by setting

$$a * b = z(a, b) = a + b + \frac{1}{2}[a, b] + \frac{1}{12}[a, a, b] + \cdots,$$
$$[a, b]_G = w(a, b) = [a, b] - \frac{1}{2}[a, a, b] + \cdots.$$

Similarly, if G is a group, then we can sometimes invert these formulae. We have that e^{x+y} is equal to e^xe^y times an infinite product of (group-) commutators in e^x and e^y, with rational exponents, and of increasing weight (cf. [15, Lemma 10.7]). More specifically, if we define h_1 by $h_1(e^x, e^y) = e^{x+y}$ then

$$\begin{aligned} h_1(g,h) = {} & gh[g,h]^{-\frac{1}{2}}[g,g,h]^{-\frac{1}{12}}[h,g,h]^{\frac{1}{12}}[g,g,g,h]^{-\frac{1}{24}}[h,h,g,h]^{\frac{1}{24}}[g,g,g,g,h]^{-\frac{19}{720}} \\ & \cdot [g,h,g,g,h]^{-\frac{1}{30}}[g,h,h,g,h]^{\frac{37}{720}}[h,g,g,g,h]^{\frac{23}{720}}[h,h,g,g,h]^{-\frac{1}{20}} \\ & \cdot [h,h,h,g,h]^{\frac{19}{720}}[g,g,g,g,g,h]^{-\frac{3}{160}}\cdots. \end{aligned}$$

We call this the *first BCH inverse formula.*

In a similar way (see [15, Lemma 10.7]) we have that $e^{[x,y]}$ is an infinite product of commutators in e^x and e^y, with rational exponents (of course, $e^{[x,y]} = e^{xy-yx}$). We define h_2 by $h_2(e^x, e^y) = e^{[x,y]}$; then we have

$$\begin{aligned} h_2(g,h) = {} & [g,h][g,g,h]^{\frac{1}{2}}[h,g,h]^{\frac{1}{2}}[g,g,g,h]^{\frac{1}{3}}[h,g,g,h]^{\frac{1}{4}}[h,h,g,h]^{\frac{1}{3}}[g,g,g,g,h]^{\frac{1}{4}} \\ & \cdot [g,h,g,g,h]^{-\frac{1}{12}}[h,g,g,g,h]^{\frac{1}{4}}[h,h,g,g,h]^{\frac{1}{6}}[h,h,h,g,h]^{\frac{1}{4}}[g,g,g,g,g,h]^{\frac{1}{5}}\cdots. \end{aligned}$$

We call this the *second BCH inverse formula.*

Now we can define a Lie plus $+$ and a Lie multiplication $[\,,]_L$ on G by setting

$$g + h = h_1(g, h) = gh[g, h]^{-\frac{1}{2}}\cdots,$$
$$[g, h]_L = h_2(g, h) = [g, h][g, g, h]^{\frac{1}{2}}\cdots.$$

Clearly there are several problems with these "formulae", the main one being that they involve infinite sums in Lie algebras and infinite products in groups. The simplest way of avoiding this problem is to insist that the groups and Lie algebras be nilpotent. If the Lie algebra is nilpotent of class c then we can truncate the formulae for group multiplication and group commutator at the weight c terms. Similarly, if the group is nilpotent of class c then we can truncate the formulae for Lie plus and Lie multiplication at the weight c terms. The other major problem is that the formulae involve multiplication of Lie algebra elements by rational scalars, and involve extracting rational roots of group elements. The simplest way of solving both these problems is to insist that the Lie algebras be nilpotent Lie algebras over $\mathbb{Q}$, and to insist that the groups be torsion free radicable nilpotent groups. This yields the Mal'cev correspondence mentioned in the introduction. To obtain the Lazard correspondence for $p > c$ between the category of nilpotent Lie rings with order p^n and nilpotency class c and the category of finite p-groups with order p^n and nilpotency class c, we observe that the denominators of the coefficients of weight k terms in the Baker–Campbell–Hausdorff formula only involve primes that are at most k. This means that if L is a nilpotent Lie ring of order p^n and class $c < p$, then we can evaluate the coefficients in the truncated formulae for group product and group commutator as integers modulo p^n. Similar considerations apply to the inverse formulae defining a Lie plus and a Lie product in a finite p-group of class $c < p$, see [15, Section 10.2].

Note that the right-normed Lie products in x and y are not linearly independent in a Lie algebra, so there is no fixed way of expressing the BCH-formulae. Similar considerations apply to the inverse formulae in a group. Furthermore, the inverse formulae in a group are sensitive to the ordering taken on the right-normed group commutators in x and y.

It is important to realize that with the Lazard correspondence we have a single set which is simultaneously a group and a Lie ring. Similarly, with the Mal'cev correspondence we have a single set, which is both a group and a Lie algebra. The group operations can be defined in terms of the Lie operations and the Lie operations can be defined in terms of the group operations. Subgroups are subalgebras, normal subgroups are ideals, the centre is the centre, the lower central series is the lower central series, the automorphism group is the automorphism group.

3. Coefficients of the BCH-formula

As in the previous section we write $e^x e^y = e^z$. The subspace $\widehat{L}$ of $\widehat{A}$ spanned by all Lie-commutators in x and y is called the space of Lie-polynomials. The surprising fact, which is the main content of the BCH-formula, is that z lies in this space.

At the basis of all our methods lies the BCH-formula. We need an expression for z as a linear combination of commutators, truncated at a previously fixed weight c. Many methods to obtain this have been proposed in the literature (for example, [3,7,21,24]). For our purposes the method outlined in [21], based on results of Goldberg [12], is excellent: it is easy to implement, and produces the formulae we want efficiently. In this section we briefly review this method. It proceeds in two steps: first we write z as a linear combination of monomials in $\widehat{A}$, then we transform that into a linear combination of commutators.

For the first step we use Goldberg's method [12] for computing the coefficients of the monomials in $z = z(x, y)$. Let $m \geqslant 1$ and $s_1, \dots, s_m$ be positive integers. Set $\Omega_x = x^{s_1} y^{s_2} \cdots (x \vee y)^{s_m}$, where $x \vee y$ is x (respectively, y) if m is odd (respectively, even). The definition of Ω_y is the same, starting with y^{s_1}. Let c_x and c_y denote the coefficients of Ω_x and Ω_y in z. For $s \geqslant 1$ define polynomials $\psi_s \in \mathbb{Q}[t]$ by

- $\psi_1 = 1$,
- $s\psi_s = \frac{\mathrm{d}}{\mathrm{d}t} t(t-1)\psi_{s-1}$, for $s \geqslant 2$.

Set $n = \sum_{i=1}^{m} s_i$, $m' = \lfloor \frac{m}{2} \rfloor$, $m'' = \lfloor \frac{m-1}{2} \rfloor$. Goldberg proved that

$$c_x = (-1)^{n-1} c_y = \int_0^1 t^{m'} (t-1)^{m''} \psi_{s_1} \cdots \psi_{s_m} \, \mathrm{d}t.$$

In practice this has proved to give a very efficient method to compute the coefficients of Ω_x, Ω_y (cf. [21]).

The second step can be carried out using the Dynkin–Specht–Wever theorem [14, Chapter V, Theorem 8], as observed in [27]. Here we describe a slight refinement of this procedure. We also write x_1 in place of x and x_2 in place of y. Let $\varphi : \widehat{A} \to \widehat{L}$ be the linear map defined by

$$\varphi(x_{i_1} \cdots x_{i_m}) = \begin{cases} [x_{i_1}, \dots, x_{i_m}] & \text{if } i_m = 1, \\ 0 & \text{if } i_m = 2. \end{cases}$$

Lemma 3.1. *Let d_x be the number of occurrences of x in $x_{i_1} \cdots x_{i_m}$. Then $\varphi([x_{i_1}, \dots, x_{i_m}]) = d_x [x_{i_1}, \dots, x_{i_m}]$.*

Proof. As in [14, Section V.4], it can be shown that $\varphi : \widehat{L} \to \widehat{L}$ is a derivation, i.e., $\varphi([a,b]) = [\varphi(a), b] + [a, \varphi(b)]$. Also note that the lemma is trivially true if $m = 1$. For $m > 1$ we get

$$\varphi\big([x_{i_1}, \dots, x_{i_m}]\big) = \big[\varphi(x_{i_1}), x_{i_2}, \dots, x_{i_m}\big] + \big[x_{i_1}, \varphi\big([x_{i_2}, \dots, x_{i_m}]\big)\big].$$

If $i_1 = 1$, then by induction the second term is $(d_x - 1)[x_{i_1}, \dots, x_{i_m}]$. If $i_1 = 2$ then the first term is zero. In both cases we get the conclusion of the lemma. $\square$

Now let $a \in \widehat{L}$, written as linear combinations of monomials $x_{i_1} \cdots x_{i_m}$. Then we discard the monomials ending in y. The others we transform into $[x_{i_1}, \dots, x_{i_m}]$ and divide by the weight in x. According to the previous lemma, this way we get an expression for a as linear combination of brackets.

4. Notation

In Section 6 we describe our approach to computing the BCH inverse formulae. It is based on a method for computing repeated commutators of e^x and e^y, which is outlined in the next section. These sections are rather technical. For this reason, in this section we summarize some of the notation that we use throughout.

We recall the definition of the algebra $\widehat{A}$ from Section 2. We also write x_1 in place of x and x_2 in place of y. Also, for $a \in \widehat{A}$, by $[\![a]\!]_t$ we denote the homogeneous component of weight t of a. For the homogeneous components of $z \in \widehat{A}$ (defined by $e^x e^y = e^z$) we also write z_t in place of $[\![z]\!]_t$.

For $n \geqslant 2$ we set

$$I_n = \big\{(i_3, \dots, i_n) \;\big|\; i_r \in \{1,2\} \text{ for } 3 \leqslant r \leqslant n\big\}.$$

When $n = 2$ this set consists of the empty sequence. We denote elements of I_n by $\bar{\imath}$ or by $\bar{k}$. We use the convention that, after introducing such an element, its components are automatically defined, e.g., the components of $\bar{k}$ are denoted $k_3, \dots, k_n$.

For $\bar{\imath} \in I_n$ we set

$$x_{\bar{\imath}} = [x_{i_n}, \dots, x_{i_3}, x_1, x_2]. \tag{1}$$

Throughout, as already said in Section 1, we use the right-normed convention for bracketed expressions. Also we define the elements $\gamma_{\bar{\imath}} \in \mathbb{Q}$ by the equation

$$z_n = \sum_{\bar{\imath} \in I_n} \gamma_{\bar{\imath}} x_{\bar{\imath}}.$$

We also write γ_0 in place of $\gamma_{()}$; so $\gamma_0 = \frac{1}{2}$.

Example 4.1. We have

$$\begin{aligned} z_5 = &-\frac{1}{720}[x,x,x,x,y] - \frac{1}{120}[x,y,x,x,y] - \frac{1}{360}[x,y,y,x,y] + \frac{1}{360}[y,x,x,x,y] \\ &+ \frac{1}{120}[y,y,x,x,y] + \frac{1}{720}[y,y,y,x,y]. \end{aligned}$$

Hence

$$\gamma_{(1,1,1)} = -\frac{1}{720}; \qquad \gamma_{(1,2,1)} = -\frac{1}{120}; \qquad \gamma_{(2,2,1)} = -\frac{1}{360};$$
$$\gamma_{(1,1,2)} = \frac{1}{360}; \qquad \gamma_{(1,2,2)} = \frac{1}{120}; \qquad \gamma_{(2,2,2)} = \frac{1}{720}.$$

Now, for a positive integer n we consider the sets J_n and S_n defined as

$$J_n = \{(j_1, \ldots, j_n) \mid j_r \geqslant 1 \text{ for } 1 \leqslant r \leqslant n\},$$

and

$$S_n = \{(s_1, \ldots, s_n) \mid 1 \leqslant s_1 < \cdots < s_n\}.$$

For the elements of J_n and S_n, that we denote by $\bar{j}$ and $\bar{s}$ respectively, we use the same convention as for the elements of I_n. Also, for $\bar{j} \in J_n$ we define

$$\delta_{\bar{j}} = \sum_{\bar{\imath} \in I_n} \varepsilon_{\bar{\imath},\bar{j}} \gamma_{\bar{\imath}},$$

where

$$\varepsilon_{\bar{\imath},\bar{j}} = \bigl((-1)^{j_2} - (-1)^{j_1}\bigr)(-1)^{i_3 j_3 + \cdots + i_n j_n}.$$

Note that $(-1)^{j_2} - (-1)^{j_1}$ is 0 if $j_1 + j_2$ is even, and it is ± 2 if $j_1 + j_2$ is odd.

Finally let $P_{t,n}$ be the set of all ordered partitions $(p_1, \ldots, p_n)$ of t with length n. For its elements, that we denote by $\bar{p}$, again we use the same convention as for the elements of I_n.

5. Computing commutators

In this section we describe a formula for computing (the homogeneous components of) a repeated commutator of e^{x_1} and e^{x_2}. For $q \geqslant 2$ and $\bar{k} \in I_q$ we define $W_{\bar{k}}$ by

$$e^{W_{\bar{k}}} = \bigl[e^{x_{k_q}}, \ldots, e^{x_{k_3}}, e^{x_1}, e^{x_2}\bigr].$$

(This is the group commutator, e.g., $[e^{x_1}, e^{x_2}] = e^{-x_1} e^{-x_2} e^{x_1} e^{x_2}$.) We also write w instead of $W_{()}$, as in Section 2.

Let $\bar{k} \in I_q$, and $\bar{k}' \in I_{q+1}$ be such that $k'_i = k_i$ for $3 \leqslant i \leqslant q$. Then

$$e^{W_{\bar{k}'}} = \bigl[e^{x_{k_{q+1}}}, e^{W_{\bar{k}}}\bigr],$$

which we will use to compute the homogeneous components of $W_{\bar{k}'}$, assuming we have those of $W_{\bar{k}}$. More precisely, we put $X_1 = x_{k_{q+1}}$ and $X_2 = W_{\bar{k}}$. Using the BCH-formula we get expressions for the homogeneous components of $Y_1, Y_2 \in \widehat{A}$, where $e^{Y_1} = e^{-X_1} e^{-X_2}$ and $e^{Y_2} = e^{X_1} e^{X_2}$. So we want to know $W = W_{\bar{k}'}$ with the property $e^W = e^{Y_1} e^{Y_2}$. Now in order to obtain an expression for the homogeneous components of W we use the BCH-formula again. This then leads to the main theorem of this section, Theorem 5.3.

The component of weight r of X_1 is given by

$$[\![X_1]\!]_r = \begin{cases} X_1, & \text{if } r = 1, \\ 0, & \text{otherwise.} \end{cases}$$

Also $[\![X_2]\!]_r = 0$ for all $r < q$.

We write $Z = Y_2$; so $e^Z = e^{X_1}e^{X_2}$. We express Z in terms of X_1 and X_2, and we let Z_n denote the homogeneous component of weight n in X_1 and X_2 (e.g., $X_1X_2X_1$ is of weight 3). Then $Z = \sum_{n\geqslant 1} Z_n$. For $\bar{\imath} \in I_n$, we define

$$[\![X_{\bar{\imath}}]\!]_r = \sum_{r_2+\cdots+r_n=r-1} \big[[\![X_{i_n}]\!]_{r_n}, \ldots, [\![X_{i_3}]\!]_{r_3}, X_1, [\![X_2]\!]_{r_2}\big].$$

Lemma 5.1. *The homogeneous component of weight r (in x_1 and x_2) of Z_n is given by*

$$[\![Z_1]\!]_r = \begin{cases} X_1, & \text{if } r = 1, \\ [\![X_2]\!]_r, & \text{otherwise}, \end{cases}$$

and, for $n > 1$

$$[\![Z_n]\!]_r = \sum_{\bar{\imath}\in I_n} \gamma_{\bar{\imath}} [\![X_{\bar{\imath}}]\!]_r.$$

Proof. An expression for Z in terms of X_1, X_2 is given by the BCH-formula. So

$$Z_1 = X_1 + X_2 \quad \Rightarrow \quad [\![Z_1]\!]_r = [\![X_1]\!]_r + [\![X_2]\!]_r.$$

Because X_2 only has monomials of weight at least $q > 1$ and $[\![X_1]\!]_r = 0$ if $r \neq 1$, the first part is clear. Let $n > 1$. From the BCH-formula we get

$$Z_n = \sum_{\bar{\imath}\in I_n} \gamma_{\bar{\imath}} X_{\bar{\imath}} \quad \Rightarrow \quad [\![Z_n]\!]_r = \Bigg[\!\!\Bigg[\sum_{\bar{\imath}\in I_n} \gamma_{\bar{\imath}} X_{\bar{\imath}} \Bigg]\!\!\Bigg]_r = \sum_{\bar{\imath}\in I_n} \gamma_{\bar{\imath}} [\![X_{\bar{\imath}}]\!]_r. \qquad \square$$

For $\bar{\jmath} \in J_n$ we put

$$Z_{\bar{\jmath}} = [Z_{j_n}, \ldots, Z_{j_1}].$$

Let $\bar{p} \in P_{t,n}$ and consider the subset of J_n defined as

$$J_{\bar{p}} = \{\bar{\jmath} \in J_n \mid j_r \leqslant p_r \text{ for } 1 \leqslant r \leqslant n \text{ and } j_1 > j_2\}.$$

For $\bar{p} \in P_{t,n}$ and $\bar{\jmath} \in J_{\bar{p}}$, we put

$$[\![Z_{\bar{\jmath}}]\!]_{\bar{p}} = \big[[\![Z_{j_n}]\!]_{p_n}, \ldots, [\![Z_{j_1}]\!]_{p_1}\big].$$

Remark 5.2. Notice that $[\![Z_n]\!]_r = 0$ if $n > 1$ and $r \geqslant n + q - 1$. In fact, the term $[\![X_{\bar{\imath}}]\!]_r$ of least weight in $\sum_{\bar{\imath}\in I_n} \gamma_{\bar{\imath}} [\![X_{\bar{\imath}}]\!]_r$ is the one with $\bar{\imath} = (1, \ldots, 1) \in I_n$, that is

$$[\![X_{(1,\ldots,1)}]\!]_r = \big[X_1, \ldots, X_1, X_1, [\![X_2]\!]_{r-n+1}\big].$$

But the weight of the homogeneous components of X_2 is at least q. Hence $[\![X_2]\!]_{r-n+1} = 0$ if $r - n + 1 < q$ and this implies that $[\![Z_n]\!]_r = 0$ for all $r < n + q - 1$. It follows also that, for $\bar{p} \in P_{t,n}$ and $\bar{\jmath} \in J_{\bar{p}}$, we have $[\![Z_{\bar{\jmath}}]\!]_{\bar{p}} = 0$ if there is an s with $p_s \neq 1$ and $j_s > p_s - q + 1$.

Theorem 5.3. *The component of weight t (in x_1, x_2) of W is given by the formula*:

$$[\![W]\!]_t = \sum_{j\geqslant 1}\bigl(1+(-1)^j\bigr)[\![Z_j]\!]_t + \sum_{n=2}^{t}\sum_{\bar p\in P_{t,n}}\sum_{\bar j\in J_{\bar p}} \delta_{\bar j}[\![Z_{\bar j}]\!]_{\bar p}. \tag{2}$$

Remark 5.4. Notice that $[\![W]\!]_t = 0$ for all $t < q+1$. Also

$$1+(-1)^j = \begin{cases} 2, & \text{if } j \text{ is even,}\\ 0, & \text{otherwise.}\end{cases}$$

So, to calculate $[\![W]\!]_t$ using (2), in the first summand it is sufficient to consider $[\![Z_j]\!]_t$ for even j. Moreover, since $(-1)^{j_2} - (-1)^{j_1} = 0$ if $j_1 + j_2$ is even, in the second summand it is sufficient to deal with all $Z_{\bar j}$ such that $j_1 + j_2$ is odd.

Proof of Theorem 5.3. Recall that $Y_1, Y_2 \in \widehat{A}$ are defined by $e^{Y_1} = e^{-X_1}e^{-X_2}$, $e^{Y_2} = e^{X_1}e^{X_2}$. Then $e^W = [e^{X_1}, e^{X_2}] = e^{Y_1}e^{Y_2}$. By the BCH-formula we have $Y_i = \sum_{j\geqslant 1}(-1)^{ij}Z_j$, for $i = 1, 2$. Again using the BCH-formula, we get $W = \sum_{n\geqslant 1} W_n$, where $W_1 = Y_1 + Y_2$ and, for $n > 1$,

$$W_n = \sum_{\bar\imath\in I_n}\gamma_{\bar\imath}Y_{\bar\imath}.$$

(Here the definition of $Y_{\bar\imath}$ is similar to the one of $x_{\bar\imath}$ in (1).)

We prove that, for $n > 1$,

$$W_n = \sum_{\substack{\bar j\in J_n\\ j_1>j_2}} \delta_{\bar j}Z_{\bar j}. \tag{3}$$

For this, we need to show that

$$Y_{\bar\imath} = \sum_{\substack{\bar j\in J_n\\ j_1>j_2}} \varepsilon_{\bar\imath,\bar j}Z_{\bar j}. \tag{4}$$

We proceed by induction on n. For $n = 2$ we have

$$[Y_1, Y_2] = \Biggl[\sum_{j\geqslant 1}(-1)^jZ_j, \sum_{j\geqslant 1}Z_j\Biggr] = \sum_{j_1>j_2\geqslant 1}\bigl((-1)^{j_2}-(-1)^{j_1}\bigr)[Z_{j_2}, Z_{j_1}].$$

So for $n = 2$ we get (4). By induction we get

$$\begin{aligned}[][Y_{i_{n+1}}, Y_{\bar\imath}] &= \Biggl[\sum_{j\geqslant 1}(-1)^{i_{n+1}j}Z_j, Y_{\bar\imath}\Biggr]\\ &= \Biggl[\sum_{j\geqslant 1}(-1)^{i_{n+1}j}Z_j, \sum_{\substack{\bar j\in J_n\\ j_1>j_2}}\varepsilon_{\bar\imath,\bar j}Z_{\bar j}\Biggr] = \sum_{\substack{\bar j\in J_{n+1}\\ j_1>j_2}}\varepsilon_{\bar\imath,\bar j}Z_{\bar j}.\end{aligned}$$

So (4) follows for all $n \geqslant 2$. Furthermore (3) is an immediate consequence of (4).

Now

$$W_1 = Y_1 + Y_2 = \sum_{j\geqslant 1}(-1)^j Z_j + \sum_{j\geqslant 1} Z_j = \sum_{j\geqslant 1}\bigl(1+(-1)^j\bigr) Z_j.$$

Therefore

$$\begin{aligned}
[\![W]\!]_t &= \Biggl[\!\!\Biggl[\sum_{n\geqslant 1} W_n\Biggr]\!\!\Biggr]_t = \Biggl[\!\!\Biggl[\sum_{j\geqslant 1}\bigl(1+(-1)^j\bigr)Z_j + \sum_{n\geqslant 2}\sum_{\substack{\bar{j}\in J_n\\ j_1>j_2}} \delta_{\bar{j}} Z_{\bar{j}}\Biggr]\!\!\Biggr]_t \\
&= \sum_{j=1}^{t}\bigl(1+(-1)^j\bigr)[\![Z_j]\!]_t + \sum_{n=2}^{t}\Biggl[\!\!\Biggl[\sum_{\substack{\bar{j}\in J_n\\ j_1>j_2}} \delta_{\bar{j}} Z_{\bar{j}}\Biggr]\!\!\Biggr]_t \\
&= \sum_{j=1}^{t}\bigl(1+(-1)^j\bigr)[\![Z_j]\!]_t + \sum_{n=2}^{t}\sum_{\bar{p}\in P_{t,n}}\sum_{\bar{j}\in J_{\bar{p}}} \delta_{\bar{j}}[\![Z_{\bar{j}}]\!]_{\bar{p}}. \qquad \square
\end{aligned}$$

Example 5.5. We calculate the series $W = W_{(1)}$, where $e^{W_{(1)}} = [e^x, e^w] = [e^x, e^x, e^y]$. We have

$$\begin{aligned}
w = {}& [x,y] - \frac{1}{2}[x,x,y] - \frac{1}{2}[y,x,y] + \frac{1}{6}[x,x,x,y] + \frac{1}{4}[y,x,x,y] + \frac{1}{6}[y,y,x,y] \\
&- \frac{1}{24}[x,x,x,x,y] - \frac{1}{12}[x,y,y,x,y] - \frac{1}{12}[y,x,x,x,y] - \frac{1}{24}[y,y,y,x,y] + \cdots.
\end{aligned}$$

Now we put $X_1 = x$ and $X_2 = w$ and $e^Z = e^x e^w$. Then we have

$$\begin{aligned}
[\![W]\!]_3 &= 2[\![Z_2]\!]_3, \\
[\![W]\!]_4 &= 2[\![Z_2]\!]_4 + \delta_{(2,1)}\bigl[[\![Z_1]\!]_1, [\![Z_2]\!]_3\bigr], \\
[\![W]\!]_5 &= 2\bigl([\![Z_2]\!]_5 + [\![Z_4]\!]_5\bigr) + \delta_{(2,1)}\bigl(\bigl[[\![Z_1]\!]_1, [\![Z_2]\!]_4\bigr] + \bigl[[\![Z_1]\!]_2, [\![Z_2]\!]_3\bigr]\bigr) \\
&\quad + \delta_{(2,1,1)}\bigl[[\![Z_1]\!]_1, [\![Z_1]\!]_1, [\![Z_2]\!]_3\bigr].
\end{aligned}$$

Now $[\![Z_1]\!]_1 = x$, $[\![Z_1]\!]_r = [\![w]\!]_r$ for all $r > 1$, $[\![Z_2]\!]_r = \gamma_0[x, [\![w]\!]_{r-1}]$ for all $r \geqslant 3$ and $[\![Z_4]\!]_5 = 0$. Hence

$$\begin{aligned}
W_{(1)} = {}& [x,x,y] - [x,x,x,y] - \frac{1}{2}[y,x,x,y] + \frac{7}{12}[x,x,x,x,y] + \frac{1}{6}[x,y,y,x,y] \\
&+ \frac{1}{2}[y,x,x,x,y] + \cdots.
\end{aligned}$$

6. The BCH inverse formulae

We consider all $e^{W_{\bar{k}}}$ for $\bar{k} \in I_{q+1}$ and $2 \leqslant q \leqslant \tilde{q}$, for a certain $\tilde{q} \geqslant 2$. We say that the *length* of such an element is $q+1$. For $q = 1$, we have $e^{W_{()}} = e^w$. We order these elements by increasing length and, inside each length, in an arbitrary but fixed way. We denote these elements ordered like this by $e^{V_2}, \ldots, e^{V_N}$, where $N = 2^{\tilde{q}} = 1 + \sum_{q=1}^{\tilde{q}} 2^{q-1}$ (because for every $q \geqslant 1$ we have 2^{q-1} elements of length $q+1$). We denote by q_s the length of e^{V_s} for $2 \leqslant s \leqslant N$. Also we set $e^{V_1} = e^Z = e^{x_1}e^{x_2}$ and we put $q_1 = 1$.

From Section 2 we recall that the first BCH inverse formula is given by h_1, where $h_1(e^{x_1}, e^{x_2}) = e^{x_1+x_2}$. We have that $h_1(e^{x_1}, e^{x_2})$ is a product of commutators in e^{x_1}, e^{x_2} of increasing weight, with rational exponents. Such a commutator is equal to an appropriate e^{V_s}. So we get h_1 from the following equation

$$e^{x_1+x_2} = \prod_{s\geqslant 1} e^{\alpha_s V_s}, \tag{5}$$

where $\alpha_s \in \mathbb{Q}$ with $\alpha_1 = 1$. In this section we describe a method for finding α_s for $2 \leqslant s \leqslant N$. The main point is that we find a formula for the homogeneous components of the right-hand side of (5), in terms of the homogeneous components of the various V_s. Since we know the homogeneous components of the left-hand side, and we have expressions for the homogeneous components of the V_s (from Theorem 5.3), this leads to equations in the α_s. Moreover, if we proceed by increasing degree, then these equations turn out to be linear.

Lemma 6.1. *We have*

$$e^{x_1+x_2} = \prod_{s\geqslant 1}\Big(1 + \sum_{t\geqslant q_s} \big[\!\big[e^{\alpha_s V_s}\big]\!\big]_t\Big),$$

where

$$\big[\!\big[e^{\alpha_s V_s}\big]\!\big]_t = \sum_{n=1}^{\lfloor t/q_s\rfloor} \frac{\alpha_s^n}{n!} \sum_{\substack{p_1+\cdots+p_n=t\\ p_1,\dots,p_n\geqslant q_s}} [\![V_s]\!]_{p_1}\cdots[\![V_s]\!]_{p_n}. \tag{6}$$

Proof. We have

$$e^{x_1+x_2} = \prod_{s\geqslant 1} e^{\alpha_s V_s} = \prod_{s\geqslant 1}\Big(1 + \sum_{t\geqslant q_s} \big[\!\big[e^{\alpha_s V_s}\big]\!\big]_t\Big).$$

Now

$$e^{\alpha_s V_s} = 1 + \sum_{n\geqslant 1} \frac{1}{n!}(\alpha_s V_s)^n \quad\Rightarrow\quad \big[\!\big[e^{\alpha_s V_s}\big]\!\big]_t = \sum_{n\geqslant 1} \frac{\alpha_s^n}{n!}\big[\!\big[V_s^n\big]\!\big]_t.$$

Furthermore,

$$\big[\!\big[V_s^n\big]\!\big]_t = \sum_{\substack{p_1+\cdots+p_n=t\\ p_1,\dots,p_n\geqslant q_s}} [\![V_s]\!]_{p_1}\cdots[\![V_s]\!]_{p_n},$$

because $[\![V_s]\!]_t = 0$ if $t < q_s$. It follows that we must have $t \geqslant nq_s$, or $n \leqslant t/q_s$. So we get (6). $\square$

For $t, n \geqslant 1$ consider the subset of $P_{t,n}$ defined as

$$P^*_{t,n} = \{\bar{p} \in P_{t,n} \mid p_r \geqslant q_r \text{ for } 1 \leqslant r \leqslant n\}.$$

Note that $P^*_{t,n} = \emptyset$ if $n > t$. For $\bar{s} \in S_n$ and $\bar{p} \in P^*_{t,n}$, we put

$$[\![e^{\alpha_{\bar{s}} V_{\bar{s}}}]\!]^*_{\bar{p}} = [\![e^{\alpha_{s_1} V_{s_1}}]\!]_{p_1} \cdots [\![e^{\alpha_{s_n} V_{s_n}}]\!]_{p_n}.$$

We observe that, if $q_r > p_r$, then $[\![e^{\alpha_{s_r} V_{s_r}}]\!]_{p_r} = 0$. So for fixed $\bar{p} \in P^*_{t,n}$ we have that $[\![e^{\alpha_{\bar{s}} V_{\bar{s}}}]\!]^*_{\bar{p}}$ is nonzero for only a finite number of $\bar{s} \in S_n$.

Theorem 6.2. *The exponents α_n in* (5)*, for $n \geqslant 2$, satisfy the equations*

$$\sum_{n=1}^{t} \sum_{\bar{s} \in S_n} \sum_{\bar{p} \in P^*_{t,n}} [\![e^{\alpha_{\bar{s}} V_{\bar{s}}}]\!]^*_{\bar{p}} = \frac{1}{t!}(x_1 + x_2)^t. \tag{7}$$

Proof. The component of weight t of $e^{x_1+x_2}$ is given by

$$[\![e^{x_1+x_2}]\!]_t = \frac{1}{t!}(x_1 + x_2)^t.$$

We prove that it is also given by

$$[\![e^{x_1+x_2}]\!]_t = \sum_{n=1}^{t} \sum_{\bar{s} \in S_n} \sum_{\bar{p} \in P^*_{t,n}} [\![e^{\alpha_{\bar{s}} V_{\bar{s}}}]\!]^*_{\bar{p}},$$

where the $[\![e^{\alpha_{s_r} V_{s_r}}]\!]_{p_r}$ are given by (6).

Using Lemma 6.1 we have that $e^{x_1+x_2} = \prod_{s \geqslant 1}(1 + \sum_{t \geqslant q_s} [\![e^{\alpha_s V_s}]\!]_t)$, which is equal to

$$1 + \sum_{n \geqslant 1} \sum_{s_1 < \cdots < s_n} \sum_{p_1 \geqslant q_1, \ldots, p_n \geqslant q_n} [\![e^{\alpha_{s_1} V_{s_1}}]\!]_{p_1} \cdots [\![e^{\alpha_{s_n} V_{s_n}}]\!]_{p_n}.$$

Hence

$$\begin{aligned} [\![e^{x_1+x_2}]\!]_t &= \left[\!\!\left[1 + \sum_{n \geqslant 1} \sum_{s_1 < \cdots < s_n} \sum_{p_1 \geqslant q_1, \ldots, p_n \geqslant q_n} [\![e^{\alpha_{s_1} V_{s_1}}]\!]_{p_1} \cdots [\![e^{\alpha_{s_n} V_{s_n}}]\!]_{p_n} \right]\!\!\right]_t \\ &= \sum_{n=1}^{t} \sum_{\bar{s} \in S_n} \sum_{\bar{p} \in P^*_{t,n}} [\![e^{\alpha_{\bar{s}} V_{\bar{s}}}]\!]_{\bar{p}}. \end{aligned}$$

Then we get the conclusion of the theorem. $\square$

Note that (7) is linear in the α_s where s is such that $q_s = t$. However, proceeding by increasing weight, we may assume that the α_s with $q_s < t$ already have been determined. So we get the α_s by solving linear equations. We illustrate this procedure by an example.

Example 6.3. We want to calculate all the exponents α_s in (5) up to weight 4. We order the commutators in the following way:

$$\begin{aligned}
&V_1 = z,\\
&V_2 = w,\\
&V_3 = W_{(1)}, \qquad V_4 = W_{(2)},\\
&V_5 = W_{(1,1)}, \qquad V_6 = W_{(2,1)}, \qquad V_7 = W_{(1,2)}, \qquad V_8 = W_{(2,2)}.
\end{aligned}$$

Since $\alpha_1 = 1$, for $t = 2$ we get

$$\frac{1}{2}(x+y)^2 = [\![V_1]\!]_2 + \alpha_2[\![V_2]\!]_2 \quad \Rightarrow \quad \frac{1}{2}(x+y)^2 = z_2 + \frac{1}{2}z_1^2 + \alpha_2[\![w]\!]_2.$$

This is the same as $(\frac{1}{2} + \alpha_2)[x, y] = 0$, whence $\alpha_2 = -\frac{1}{2}$. For $t = 3$ the equation becomes

$$\frac{1}{6}(x+y)^3 = [\![V_1]\!]_3 - \frac{1}{2}[\![V_2]\!]_3 + \alpha_3[\![V_3]\!]_3 + \alpha_4[\![V_4]\!]_3 - \frac{1}{2}[\![V_1]\!]_1[\![V_2]\!]_2,$$

that is

$$\frac{1}{6}(x+y)^3 = z_3 + \frac{1}{2}(z_1z_2 + z_2z_1) + \frac{1}{6}z_1^3 - \frac{1}{2}[\![w]\!]_3 + \alpha_3[\![W_{(1)}]\!]_3 + \alpha_4[\![W_{(2)}]\!]_3 - \frac{1}{2}z_1[\![w]\!]_2,$$

and after expanding we find $\alpha_3 = -\frac{1}{12}$ and $\alpha_4 = \frac{1}{12}$. Proceeding in the same way, for $t = 4$ we obtain the equations $\alpha_5 = -\frac{1}{24}$, $\alpha_6 + \alpha_7 = 0$ and $\alpha_8 = \frac{1}{24}$. So we see that the solution to these equations is not uniquely determined. We choose $\alpha_6 = \alpha_7 = 0$. Then the series e^{x+y} becomes

$$e^{x+y} = e^z e^{-\frac{1}{2}w} e^{-\frac{1}{12}W_{(1)}} e^{\frac{1}{12}W_{(2)}} e^{-\frac{1}{24}W_{(1,1)}} e^{\frac{1}{24}W_{(2,2)}} \cdots.$$

Now we consider the second BCH inverse formula, given by $h_2(e^{x_1}, e^{x_2}) = e^{[x_1,x_2]}$ (see Section 2). Analogously to h_1, we get h_2 from the equality

$$e^{[x_1,x_2]} = \prod_{s \geqslant 2} e^{\beta_s V_s}, \tag{8}$$

where $\beta_s \in \mathbb{Q}$ with $\beta_2 = 1$. Again the goal is to find the β_s for all $s \geqslant 3$. The proof of the next theorem is the same as the one for Theorem 6.2.

Theorem 6.4. *The exponents β_s in* (8)*, for $s \geqslant 3$, satisfy the equations*

$$\sum_{n=1}^{t} \sum_{\bar{s} \in S_n} \sum_{\bar{p} \in P_{t,n}^*} [\![e^{\beta_{\bar{s}} V_{\bar{s}}}]\!]_{\bar{p}}^* = \begin{cases} \frac{1}{(\frac{t}{2})!}[x_1, x_2]^{\frac{t}{2}}, & \textit{if } t \textit{ is even},\\ 0, & \textit{otherwise}. \end{cases}$$

7. Setting up the correspondence

Let G be a p-group of nilpotency class smaller than p. In this section we show how to compute the Lie ring structure of G. From Section 2 we recall that this Lie ring structure is defined by $g + h = h_1(g, h)$ and $[g, h]_L = h_2(g, h)$. Moreover, in the previous sections we have shown how to compute formulae for h_1 and h_2 that enable us to evaluate $h_1(g, h)$ and $h_2(g, h)$ for all $g, h \in G$.

We assume that G is given by a consistent power-commutator presentation (cf. [26]). This means that we have generators $g_1, \ldots, g_n$ of G together with relations of the form

$$g_i^p = g_{i+1}^{d_{i+1}^{(i)}} \cdots g_n^{d_n^{(i)}} \quad \text{where } 1 \leqslant i \leqslant n \text{ and } d_j^{(i)} < p,$$

$$[g_j, g_i]_G = g_{j+1}^{c_{j+1}^{(i,j)}} \cdots g_n^{c_n^{(i,j)}} \quad \text{where } 1 \leqslant i < j \leqslant n \text{ and } c_k^{(i,j)} < p.$$

These relations can be used to write every element of G as a normal word $g_1^{e_1} \cdots g_n^{e_n}$ where $0 \leqslant e_i < p$. The presentation is called *consistent* if G has order p^n, or equivalently, if different normal words give different elements of G. In a group given by a consistent power-commutator presentation, the collection algorithm (cf. [16,26,28]) can be used to compute the normal word representing the product of two elements.

Now every $g \in G$ can also be expressed as a sum $\sum_{i=1}^n \alpha_i g_i$, with $\alpha_i \in \mathbb{Z}$. Indeed, write $g = g_1^{e_1} \cdots g_n^{e_n}$. Then $g = e_1 g_1 + R$, with

$$R = -e_1 g_1 + g = g_1^{-e_1} + g = h_1\big(g_1^{-e_1}, g\big) = g_1^{-e_1} g S,$$

where S lies in $[G,G]$. But $[G,G]$ is contained in the subgroup generated by $g_3, \dots, g_n$. So R lies in the subgroup generated by $g_2, \dots, g_n$. Hence, by induction, $R = \sum_{i=2}^n \alpha_i g_i$, where $\alpha_i \in \mathbb{Z}$. Setting $\alpha_1 = e_1$ we get $g = \sum_{i=1}^n \alpha_i g_i$. In particular this implies that G, as a Lie ring, is also generated by $g_1, \dots, g_n$.

We note that this argument also yields an algorithm for computing the α_i, given the exponents e_i.

In order to compute the Lie ring structure we transform the power-commutator relations of G into relations that hold in the Lie ring. That is, we compute $[g_j, g_i]_L = h_2(g_j, g_i)$ which is then transformed to a sum of the form $\sum_{i=1}^n \alpha_i g_i$, with $\alpha_i \in \mathbb{Z}$. Similarly we compute $pg_i = g_i^p$, and transform it to a sum of the same form. Using these we can compute the Lie commutator of any two elements of G, and write it as a sum of generators $\sum_{i=1}^n \beta_i g_i$ with $0 \leqslant \beta_i < p$.

Once we have the Lie ring structure on G, we have two representations of $g \in G$. The first is as a product $g = g_1^{e_1} \cdots g_n^{e_n}$, which we call the product representation. The second is as a sum $g = \sum_{i=1}^n \alpha_i g_i$, which we call the sum representation. It is important to note that both representations give the same element. The different representations reflect the different operations that are used to express the element in terms of the generators.

We can use the BCH-formula to efficiently switch between representations. As in Section 2 we write $g * h = z(g,h)$ for the result of evaluating the BCH-formula in g and h. This evaluation uses only the operations of the Lie ring structure on G. By the Lazard correspondence we have that $g * h = gh$ (i.e., the result of evaluating the BCH-formula in g, h is equal to the product in G of g, h). Let $g \in G$ be given in the product representation, $g = g_1^{e_1} \cdots g_n^{e_n}$. Then $g = g_1^{e_1} * \cdots * g_n^{e_n} = (e_1 g_1) * \cdots * (e_n g_n)$. Since evaluating the BCH-formula uses Lie ring operations only, the result of this is the sum representation of g. Conversely, suppose that g is given as $g = \sum_{i=1}^n \alpha_i g_i$. Then we write $g = g_1^{\alpha_1} P$, where $P = (-\alpha_1 g_1) * g$. By evaluating the BCH-formula we get $P = \sum_{i=2}^n \beta_i g_i$, and we continue with P. At the end we obtain the product representation of g.

We can also perform these operations symbolically, since it is possible to evaluate $z(\sum_i x_i g_i, \sum_i y_i g_i)$, where the x_i and y_i are indeterminates of a polynomial ring over $\mathbb{Q}$. In this way we obtain polynomials $p_1, \dots, p_n$ in the indeterminates x_i, y_i such that

$$z\left(\sum_{i=1}^n \alpha_i g_i, \sum_{i=1}^n \beta_i g_i\right) = \sum_{i=1}^n p_i(\alpha_1, \dots, \alpha_n, \beta_1, \dots, \beta_n) g_i.$$

Since the operations of switching between the different representations involve the BCH-formula only, we can do the same for them. That is, we can obtain polynomials $f_1, \dots, f_n$, $t_1, \dots, t_n$ that depend on n indeterminates, such that

$$\sum_{i=1}^{n} f_i(e_1, \dots, e_n) g_i$$

is the sum representation of $g_1^{e_1} \cdots g_n^{e_n}$, and

$$g_1^{t_1(\alpha_1,\dots,\alpha_n)} \cdots g_n^{t_n(\alpha_1,\dots,\alpha_n)}$$

is the product representation of $\sum_i \alpha_i g_i$.

Remark 7.1. The term *T-group* is short for a finitely-generated torsion-free nilpotent group (cf. [25, §3.C]). Let G be a T-group, and we assume that it is given by a consistent power-commutator presentation. This works in the same way as for p-groups, except that there are no power relations. In other words, we have generators $g_1, \dots, g_n$ and relations

$$[g_j, g_i]_G = g_{j+1}^{c_{j+1}^{(i,j)}} \cdots g_n^{c_n^{(i,j)}} \quad \text{where } 1 \leqslant i < j \leqslant n \text{ and } c_k^{(i,j)} \in \mathbb{Z}.$$

In this case every element of the group can be written as a unique normal word $g_1^{e_1} \cdots g_n^{e_n}$, where $e_i \in \mathbb{Z}$.

A group $\widetilde{G}$ is called *radicable* if for all $\tilde{g} \in \widetilde{G}$ and $n \in \mathbb{Z}$ there exists a (necessarily unique) $\tilde{h} \in \widetilde{G}$ such that $\tilde{h}^n = g$ (cf. [25, §6.A]). For a T-group G there exists a unique minimal radicable group $\widetilde{G}$ containing G [25, §6.A]. This is called the *radicable hull* of G.

For a nilpotent radicable group $\widetilde{G}$ we can use the inverse BCH formulae to define a structure of a Lie algebra over $\mathbb{Q}$ on $\widetilde{G}$. This way we get a correspondence between nilpotent radicable groups and nilpotent Lie algebras over $\mathbb{Q}$. This is called the *Mal'cev* correspondence (see [15, §10.1]). If we let $\widetilde{G}$ be the radicable hull of G, then the corresponding Lie algebra has dimension n, and is spanned by $g_1, \dots, g_n$. The multiplication table of the Lie algebra can be computed using the commutator relations above. However, in this case some care is needed. In the formula

$$[g, h]_L = [g, h]_G [g, g, h]_G^{\frac{1}{2}} [h, g, h]_G^{\frac{1}{2}} \cdots,$$

we have $g, h \in G$, whereas, for example, $[g, g, h]_G^{\frac{1}{2}} \in \widetilde{G}$. But we have no way of expressing that element. We note, however, that all those elements lie in the radicable hull of the derived group $[G, G]$. By recursion we may assume that we have already constructed the Lie algebra corresponding to that group. This Lie algebra is a subalgebra of the algebra that we are constructing. Therefore, by using the BCH-formula, we can evaluate the right-hand side of the above expression in the Lie algebra of the radicable hull of $[G, G]$. It follows that we can construct the Lie algebra corresponding to $\widetilde{G}$. This is similar to one of the approaches used in [2] to obtain the Lie algebra of the radicable hull of a T-group. However, in that paper a different method to compute the brackets $[g, h]_L$ is used.

8. Implementation and practical experiences

We have implemented the algorithms described in this paper in the language of the computer algebra system Magma V2.17 [4]. All running times reported in this section have been obtained on a 3.16 GHz processor.

To start we first compute the BCH-formula, and formulae for h_1 and h_2, up to a previously fixed weight c. For computing the BCH-formula we use the method outlined in Section 3, whereas for h_1 and h_2 we use the algorithms described in Section 6. We computed the formulae for $c = 12, 13, 14$; the running times are displayed in Table 1.

From Table 1 we see that the number of terms of the formulae, unsurprisingly, roughly doubles with each increase of the weight. However, the running times for h_1, h_2 much more than double. So

Table 1
Running times (in seconds) for the algorithms to compute the BCH-formula and formulae for h_1 and h_2. For each formula the number of terms is also given. The first column has the weight c up to which we compute the formulae.

c	h_1		h_2		BCH	
	time	# terms	time	# terms	time	# terms
12	526	1519	433	1517	0.13	985
13	2329	3055	2013	3053	0.47	2521
14	11 137	6111	12 493	6109	0.92	4056

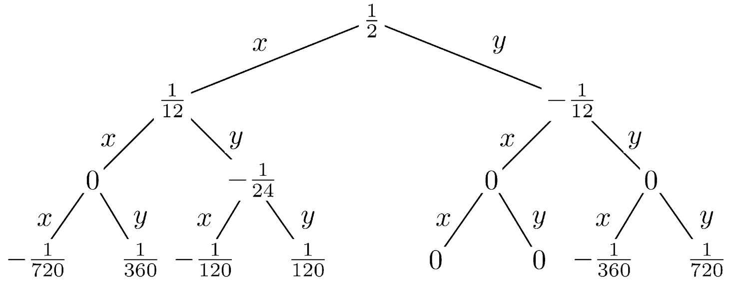

Fig. 1. Tree corresponding to the part of the BCH-formula up to weight 5.

it will be possible to go a bit further, until $c = 15$, or even $c = 16$. But we cannot realistically hope to go much beyond that. It is also seen that the algorithm for computing the BCH-formula is much more efficient than the algorithms for computing its inverses. For this reason it is possible to compute the BCH-formula up to a higher weight (like $c = 20$).

Currently, our programs use the formulae until weight 14, which are stored in a file. This means that currently our programs are able to cope with groups of classes up to 14.

The main operation of our algorithms is the evaluation of the BCH-formula for given elements x, y of a nilpotent Lie ring, and the formulae for h_1 and h_2 for given elements g, h of a p-group. In order to do this efficiently, we encode these formulae as labeled binary trees. We describe how the tree is defined for the BCH-formula, $z(x, y) = x + y + \frac{1}{2}[x, y] + \cdots$. The edges of the tree are labeled x or y. And every node corresponds to a (right-normed) commutator. The root of the tree corresponds to $[x, y]$. In order to determine the commutator corresponding to any other node, we take the path to the root, and record the labels. Suppose that the labels that we encounter are $x_{i_1}, \ldots, x_{i_k}$, where x_{i_1} is the label of the edge closest to the node we are considering, and x_{i_k} is the label of the last edge, connected to the root. Then the corresponding commutator is

$$[x_{i_1}, \ldots, x_{i_k}, x, y].$$

Finally, every node has a label, which is the coefficient of the corresponding commutator in the BCH-formula. Note that this means that some of the nodes have label 0. Fig. 1 displays the tree corresponding to the BCH-formula up to weight 5.

Let T_c denote the tree corresponding to the BCH-formula, containing the terms up to weight c. In order to evaluate the BCH-formula for given x, y in a Lie ring of nilpotency class $\leqslant c$, we traverse T_c "breadth first". That is, we loop through all the nodes of depth k, and after that through all the nodes of depth $k + 1$, and so on. Every node corresponds to an element of the Lie ring, called its value, which is the corresponding commutator evaluated in x and y. When looping through the nodes of depth $k + 1$ we use the stored values corresponding to the nodes of depth k. So for every new value we need to perform one multiplication in the Lie ring.

Initially we set $z_0 = x + y$. For every node that we encounter, with label c and value v, we add cv to z_0. Then after having traversed the entire tree we have $z_0 = z(x, y)$. This approach is very efficient

Table 2
Running times (in seconds) for the algorithm to construct a Lie ring of a p-group via the Lazard correspondence. The first column displays the prime and the second has the class. The third and fourth columns have respectively the size of G_p^c and the time to construct its Lie ring. The fifth and sixth columns have the same data for H_p^c.

p	c	$\|G_p^c\|$	time	$\|H_p^c\|$	time
17	13	p^{38}	0.2	p^{37}	0.15
211	13	p^{38}	2.4	p^{37}	0.18
17	14	p^{49}	0.4	p^{48}	0.27
211	14	p^{49}	5.7	p^{48}	0.37

for two reasons: every new value comes at the cost of performing one multiplication in the Lie ring, and if a value turns out to be 0, then we can discard the entire subtree below it.

Remark 8.1. This approach also works for evaluating h_1 and h_2. However, for these two formulae some care is needed when traversing the tree: we must make sure that the commutators appear in the same order as in the formulae.

Now in order to test the algorithm for computing the Lie ring corresponding to a p-group (of class $< p$) we use the following sample groups. For a prime p let

$$\mathcal{G}_p = \langle a, b \mid [a, b, b, a, b], [a, a, b], a^{p^2}, b^p \rangle,$$

and we let G_p^c be the p-quotient of class c of $\mathcal{G}_p$. Also, let

$$\mathcal{H}_p = \langle a, b \mid [b, a, a, a, b], [b, a, b], a^p, b^p \rangle,$$

and we let H_p^c be the p-quotient of class c of $\mathcal{H}_p$.

We have constructed the Lie rings corresponding to G_p^c and H_p^c for $c = 13, 14$ and $p = 17, 211$. The running times that we obtained are shown in Table 2.

We see that it is no problem at all to construct the Lie ring of a p-group of class 13 or 14. The running times for the construction of the Lie rings of G_p^c and H_p^c are nearly equal for small primes. However, for large primes the construction of the Lie ring of G_p^c needs markedly more time. We believe that this is due to the fact that in G_p^c for large p the collection algorithm, used for multiplying elements, on the average takes more time than in H_p^c for large p.

9. Applications

In this section we briefly discuss two applications of the effective version of the Lazard correspondence.

9.1. Hall polynomials

We observe that the Lazard correspondence can be used to multiply two elements g, h (given in product representation) in a p-group G. Indeed, for this we first get the sum representations of g and h, then compute $g * h$, and return the product representation of that element.

Alternatively, we can compute the polynomials p_i, f_i, t_i defined at the end of Section 7, and substitute them into each other, in the obvious way, to get polynomials $q_1, \dots, q_n$, depending on $2n$ indeterminates, such that after evaluating

$$v_i = q_i(e_1, \dots, e_n, d_1, \dots, d_n),$$

Table 3
Running times (in seconds) for the algorithms for computing the Hall-polynomials. The first column has the group, the second and third, respectively, the running times for computing the Lazard polynomials using the algorithm based on the Lazard correspondence, and the Deep-Thought algorithm.

group	Lazard	DT
G_{31}^{14}	65.5	74.2
G_{101}^{14}	66.4	71.5
H_{31}^{14}	69.6	59.2
H_{101}^{14}	65.1	59.6

we get that

$$g_1^{e_1} \cdots g_n^{e_n} \cdot g_1^{d_1} \cdots g_n^{d_n} = g_1^{v_1} \cdots g_n^{v_n}.$$

In [13], P. Hall showed the existence of such polynomials; for this reason they are called *Hall-polynomials*. We conclude that for p-groups of class $< p$, the effective version of the Lazard correspondence yields an algorithm to compute Hall-polynomials. We have implemented this algorithm in MAGMA.

The coefficients of the q_i lie in $\mathbb{Q}$. Alternatively, if the exponent of G is p^k, then we can coerce the coefficients into $\mathbb{Z}/p^k\mathbb{Z}$. If $k > 1$ then it may happen that some of the v_i are greater than $p - 1$. In that case some further operations have to be performed to get the exponents of the normal word representing the product. In our implementation these operations are simply performed by the built-in MAGMA collector. The extra collections needed are quite simple, and take virtually no time. Also, if the group has exponent p, then the coefficients of the polynomials can be coerced into $\mathbb{F}_p$. In that case they directly give the exponents of the normal word.

Leedham-Green and Soicher [17] developed an algorithm, called *Deep-Thought*, for computing Hall-polynomials for any finitely-generated nilpotent group. They also showed how the extra operations, to get all exponents smaller than p, can be performed with polynomials. This algorithm has been implemented in GAP by Merkwitz [19]. The current version of GAP [10] contains this implementation (although it does not appear to be documented). We have compared the running times of this implementation and ours for the groups G_p^c and H_p^c, for $c = 14$ and $p = 31, 101$. The results are displayed in Table 3. The two algorithms are implemented in different systems: the Deep-Thought algorithm in GAP is partly implemented in the kernel, and partly in the GAP language, whereas the Lazard correspondence is implemented entirely in the MAGMA language. The raw timing figures look pretty comparable, but it is difficult to draw any conclusions since the two programs are running on different systems.

Remark 9.1. We have also performed experiments with the Hall-polynomials, using them for doing multiplications in p-groups. For this we have taken a thousand pairs of random elements in groups G_p^{14} and H_p^{14}, for various p, and compared the running times of the multiplications using the Hall-polynomials and using the built-in MAGMA collector. It turned out that the time needed for multiplication using the Hall-polynomials was roughly constant for all primes. This is no surprise, as a multiplication boils down to evaluating a set of polynomials at a point. However, the built-in MAGMA collector needs almost no time when the prime is small (e.g., $p = 31$), but when the prime increases it needs more time. This is to be expected because the average exponents of the random words get bigger. It turned out that the cross-over point, where the Hall-polynomials start beating the MAGMA collector is around primes of the order of magnitude of 150. For example, a thousand random multiplications in G_{101}^{14} took 95.5 seconds with the Hall-polynomials, and 8.9 seconds with the MAGMA collector. But in G_{211}^{14} the respective timings were 96.0 seconds and 213.9 seconds. We conclude that for small primes p it is better to use collection for doing multiplications in p-groups. However, when

p gets bigger, the extra effort to compute the Hall-polynomials does pay off. Furthermore, this comparison depends also on the efficiency of the implementation of the collection algorithm. The collector for p-groups in Magma is very efficient, and coded in the Magma C-kernel, whereas our programs are written in the Magma programming language, which inherently makes them a bit less efficient. Experiments reported in [17] (partly taken from the diploma thesis of Merkwitz [19]) are more positive for the approach with the Hall-polynomials. We believe that in those experiments the multiplication with polynomials was tried against a much less optimized collector. Also, we remark that the polynomials can be used for other purposes as well (see below for an example), not just for doing multiplication.

9.2. Faithful modules

As a second application we consider the problem of computing a faithful matrix representation. Let G be a p-group of exponent p and class $c < p$. Then the corresponding Lie ring is a Lie algebra over $\mathbb{F}_p$. Let $\rho : G \to M_s(\mathbb{F}_p)$ (the set of $s \times s$-matrices over $\mathbb{F}_p$) be a faithful representation of G as a Lie algebra. Suppose that $\rho(g)^c = 0$ for all $g \in G$. (We note that such a representation can be computed using the methods of [6].) Then we can define $\rho_e : G \to M_s(\mathbb{F}_p)$ by $\rho_e(g) = \exp \rho(g)$. By the BCH-formula, and the Lazard correspondence, this gives a faithful representation of G as a group.

The same approach works for a T-group. Nickel has given a very efficient method for computing a faithful representation of such a group in [22]. Briefly, this works as follows. Let G be a T-group, with polycyclic generators $g_1, \dots, g_n$. Then G acts on the dual $(\mathbb{Q}G)^*$ by $g \cdot f(a) = f(g^{-1}a)$. Now consider the linear functions $t_i \in (\mathbb{Q}G)^*$ given by $t_i(g_1^{e_1} \cdots g_n^{e_n}) = e_i$. Nickel proved that the G-submodule M of $(\mathbb{Q}G)^*$ generated by $t_1, \dots, t_n$ is faithful and finite-dimensional. In order to compute a basis of it, note that there are polynomials $q_{i,j} \in \mathbb{Q}[x_1, \dots, x_n]$ such that

$$g_j^{-1} g_1^{e_1} \cdots g_n^{e_n} = g_1^{q_{1,j}(e_1,\dots,e_n)} \cdots g_n^{q_{n,j}(e_1,\dots,e_n)}. \tag{9}$$

Moreover, $R = \mathbb{Q}[x_1, \dots, x_n]$ can be viewed as a subspace of $(\mathbb{Q}G)^*$ by identifying a polynomial h with the linear function that maps $g^{e_1} \cdots g_n^{e_n}$ to $h(e_1, \dots, e_n)$. With this identification $g_j \cdot h = h(q_{1,j}, \dots, q_{n,j})$. So R is a G-submodule of $(\mathbb{Q}G)^*$. Furthermore, the module M defined above is contained in R. Hence a basis of M can be computed by repeatedly substituting the $q_{i,j}$ into other polynomials. This makes this method very efficient. It can also be applied to p-groups of exponent p as in that case the associated Lie ring is in fact a Lie algebra over $\mathbb{F}_p$. It is doubtful whether a method based on constructing the Lie algebra first, and then constructing a module of that, will be more efficient. Here we will not investigate that. Instead, we note that it is also possible to use the inverse route and get an algorithm for computing a faithful module of a nilpotent Lie algebra L over $\mathbb{Q}$. Indeed, the BCH-formula defines the structure of a radicable nilpotent group on L. Take a basis $g_1, \dots, g_n$ of L such that $g_{i_k}, \dots, g_n$ is a basis of the k-th term of the lower central series of L, and $i_1 < i_2 < \cdots < i_c$. Viewing L as a group, we get commutator relations

$$[g_j, g_i]_G = g_{j+1}^{c_{j+1}^{(i,j)}} \cdots g_n^{c_n^{(i,j)}} \quad \text{where } 1 \leqslant i < j \leqslant n \text{ and } c_k^{(i,j)} \in \mathbb{Q}.$$

Furthermore, using the BCH-formula, we can compute polynomials $q_{i,j}$ with (9). Using these, we compute a basis of the module M, generated by the functions t_i. This yields a (group-) representation $\rho : L \to \mathrm{GL}(d, \mathbb{Q})$, where $d = \dim M$. For $g \in L$ the matrix $\rho(g)$ is unipotent. We define ρ_l by $\rho_l(g) = \log(\rho(g))$; then ρ_l is a faithful (Lie algebra-) representation of L.

In fact, if we construct the basis of L in such a way that it contains a basis of the centre of L, then it is enough to take the t_i that correspond to the basis elements of the centre, as generators of the module. Indeed, then the centre will act faithfully, and that implies that the whole Lie algebra acts faithfully. This, in most cases, leads to a smaller dimensional module.

Table 4
Running times (in seconds) for the algorithms "Mal'cev" and "Dual" for constructing a faithful representation of a nilpotent Lie algebra. The first column has the Lie algebra, the second and third columns its dimension and class. The next two columns have the running time and dimension of the computed module for the algorithm "Mal'cev". The last two columns have this data for the algorithm "Dual".

L	dim	class	Mal'cev		Dual	
			time	dim	time	dim
$\mathfrak{f}_{13}$	13	12	1.4	43	7.9	43
$\mathfrak{f}_{14}$	14	13	3.0	53	16.0	53
$\mathfrak{f}_{15}$	15	14	7.2	64	34.5	64
$N_{2,9}$	127	9	62.6	269	75.6	214
$N_{3,6}$	196	6	76.3	289	170.8	296
$N_{4,5}$	294	5	314.8	357	366.0	400
$N_{5,4}$	205	4	91.2	244	103.7	251

In Table 4 we collect some experimental data regarding this algorithm. The algorithm "Mal'cev" is the one described above, whereas "Dual" is one of the algorithms considered in [6]. The Lie algebras $\mathfrak{f}_n$ are described in [6], and $N_{m,c}$ is the free nilpotent Lie algebra with m generators, of nilpotency class c. The algorithm "Mal'cev" is faster on all examples. However, for some inputs "Dual" yields a module of smaller dimension. On other inputs "Mal'cev" also wins in this respect.

References

[1] Björn Assmann, Guarana. A GAP4 package, http://www.gap-system.org/Packages/guarana.html, 2007.
[2] Björn Assmann, Stephen Linton, Using the Mal'cev correspondence for collection in polycyclic groups, J. Algebra 316 (2) (2007) 828–848.
[3] Asok Bose, Dynkin's method of computing the terms of the Baker–Campbell–Hausdorff series, J. Math. Phys. 30 (9) (1989) 2035–2037.
[4] W. Bosma, J. Cannon, C. Playoust, The Magma algebra system. I. The user language, in: Computational Algebra and Number Theory, London, 1993, J. Symbolic Comput. 24 (3–4) (1997) 235–265.
[5] N. Bourbaki, Groupes et Algèbres de Lie, Chapitre II, Hermann, Paris, 1972.
[6] Dietrich Burde, Bettina Eick, Willem de Graaf, Computing faithful representations for nilpotent Lie algebras, J. Algebra 322 (3) (2009) 602–612.
[7] Fernando Casas, Ander Murua, An efficient algorithm for computing the Baker–Campbell–Hausdorff series and some of its applications, J. Math. Phys. 50 (3) (2009), 033513, 23 pp.
[8] Serena Cicalò, Willem A. de Graaf, LieRing. A GAP package, http://science.unitn.it/~degraaf/liering.html, 2011.
[9] Anton Evseev, Higman's PORC conjecture for a family of groups, Bull. Lond. Math. Soc. 40 (3) (2008) 415–431.
[10] The GAP Group, GAP – groups, algorithms, and programming, version 4.4, http://www.gap-system.org, 2004.
[11] G. Glauberman, A partial extension of Lazard's correspondence for finite p-groups, Groups Geom. Dyn. 1 (4) (2007) 421–468.
[12] Karl Goldberg, The formal power series for $\log e^x e^y$, Duke Math. J. 23 (1956) 13–21.
[13] P. Hall, Nilpotent Groups: Notes of Lectures Given at the Canadian Mathematical Congress, University of Alberta, 1957.
[14] N. Jacobson, Lie Algebras, Dover, New York, 1979.
[15] E.I. Khukhro, p-Automorphisms of Finite p-Groups, London Math. Soc. Lecture Note Ser., vol. 246, Cambridge University Press, Cambridge, 1998.
[16] C.R. Leedham-Green, L.H. Soicher, Collection from the left and other strategies, J. Symbolic Comput. 9 (1990) 665–675.
[17] C.R. Leedham-Green, Leonard H. Soicher, Symbolic collection using deep thought, LMS J. Comput. Math. 1 (1998) 9–24 (electronic).
[18] A.I. Mal'cev, On some classes of infinite soluble groups, Mat. Sb. (N.S.) 28 (70) (1951) 567–588.
[19] Wolfgang W. Merkwitz, Symbolische Multiplikation in nilpotenten Gruppen mit Deep-Thought, Master's thesis, RWTH Aachen, 1997.
[20] M.F. Newman, E.A. O'Brien, M.R. Vaughan-Lee, Groups and nilpotent Lie rings whose order is the sixth power of a prime, J. Algebra 278 (1) (2004) 383–401.
[21] Morris Newman, Robert C. Thompson, Numerical values of Goldberg's coefficients in the series for $\log(e^x e^y)$, Math. Comp. 48 (177) (1987) 265–271, with microfiche supplement.
[22] Werner Nickel, Matrix representations for torsion-free nilpotent groups by deep thought, J. Algebra 300 (1) (2006) 376–383.
[23] E.A. O'Brien, M.R. Vaughan-Lee, The groups with order p^7 for odd prime p, J. Algebra 292 (1) (2005) 243–258.
[24] Matthias W. Reinsch, A simple expression for the terms in the Baker–Campbell–Hausdorff series, J. Math. Phys. 41 (4) (2000) 2434–2442.
[25] D. Segal, Polycyclic Groups, Cambridge University Press, 1994.
[26] C.C. Sims, Computation with Finitely Presented Groups, Cambridge University Press, Cambridge, 1994.

[27] Robert C. Thompson, Cyclic relations and the Goldberg coefficients in the Campbell–Baker–Hausdorff formula, Proc. Amer. Math. Soc. 86 (1) (1982) 12–14.
[28] M.R. Vaughan-Lee, Collection from the left, J. Symbolic Comput. 9 (1990) 725–733.
[29] Michael Vaughan-Lee, The Restricted Burnside Problem, second ed., London Math. Soc. Monogr. Ser. (N.S.), vol. 8, Clarendon Press/Oxford University Press, New York, 1993.