J. Symbolic Computation (1990) **9**, 665–675

Collection from the Left and Other Strategies

C. R. LEEDHAM-GREEN AND L. H. SOICHER

School of Mathematical Sciences, Queen Mary and Westfield College, Mile End Road, London E1 4NS, UK

Dedicated to G. E. Wall, in appreciation of past kindnesses, and with expectations for the future

(*Received* 4 *January* 1989)

We describe experiments with various collection strategies for the multiplication of elements of a p-group. We conclude that collection from the left is a very good strategy, then analyse the computational complexity of collection from the left, and compare it with that of collection from the right. Collection from the left also appears to be an excellent strategy for the multiplication of elements of a finite soluble group.

1. Introduction

Any finite soluble group G has a *power conjugate presentation*:

$$\langle a_1, \ldots, a_n | a_i^{p_i} = w_{ii}\,(1 \le i \le n),\ a_j a_i = a_i w_{ij}\,(1 \le i < j \le n)\rangle, \tag{1}$$

such that each p_i is a positive integer, and w_{ij} $(i \le j)$ is a normal word in $a_{i+1}, \ldots, a_n$. A *normal word* is a word of the form $a_1^{k_1} \cdots a_n^{k_n}$, with $0 \le k_i < p_i$. Any element of G can be represented by a normal word, and if each element of G is represented by a unique normal word then we say that (1) is *consistent*. If G is nilpotent then w_{ij} $(i < j)$ can be assumed to be of the form $a_j c_{ij}$, where c_{ij} is a normal word in $a_{j+1}, \ldots, a_n$, and in this case the relations $a_j a_i = a_i w_{ij}$ are usually written as $[a_j, a_i] = c_{ij}$ and (1) is called a *power commutator* (*pc*) *presentation*.

If w is any word in $a_1, \ldots, a_n$ (with positive exponents), then w can be transformed into a normal word representing the same element of G by the process of *collection*, detailed in Fig. 1. Collection consists of applying the relations of (1), which give normal words to substitute for the minimal non-normal words in $a_1, \ldots, a_n$.

It can be shown that the collection process must eventually terminate (see Sims, 1987). However, from an efficiency point of view, the choices of m in step (2) are very important. Following the theoretical work of Hall (1934), some early computer implementations of collection used *collection to the left*, in which the choice of m in step (2) is the leftmost minimal non-normal subword amongst those involving an a_i with the least possible i. By 1961, Neubüser had discovered that, rather than use collection to the left, it is much more efficient with computer space and time always to choose m to be the rightmost minimal non-normal subword (unaware of Neubüser's work, this fact was rediscovered by Wamsley around 1972). This is called *collection from the right*, and implementations for collection from the right can be found in Felsch (1976) and Havas & Nicholson (1976), to which the reader is also referred for detailed historical perspectives on the development of collection algorithms.

0747–7171/90/050665+11 $03.00/0

(1) If w is a normal word then stop.
(2) Choose any minimal non-normal subword m of w, and *collect* m, that is, transform w by replacing m by the right-hand side of the relation in (1) whose left-hand side is m. Goto step (1).

Fig. 1. Collection of w.

We decided to experiment with collection to try to find better choices for m in step (2), or at least to convince ourselves that collection from the right was right! We programmed experimental collectors which could keep counts of the number of times any particular m was replaced during a collection. To our surprise we found that always choosing m to be the leftmost minimal non-normal subword, i.e. *collection from the left* is usually much superior to collection from the right.

The following heuristic argument indicates why collection from the left is often so much better than collection from the right. Suppose we are given the presentation (1) and (for simplicity) we are multiplying

$$a_1 a_2 \cdots a_n \, a_1 a_2 \cdots a_n.$$

Collection from the left starts by moving the right hand (occurrence of) a_1 into its final position, and then cleans up the word preceding the right hand a_2 into normal form before moving the right hand a_2 into position. With collection from the right, the right hand a_1 and a_2 just sit there for most of the calculation, and all conjugates created when generators are moved across that a_1 must also move across that a_2 (and so on). The point with collection from the left is that, when a generator is moved, it is moved through a normal word into position. Later in this paper we give an analysis of the computational complexity of collection from the left for p-groups. Other collection strategies may be asymptotically better than collection from the left, but it is doubtful that there is a better strategy that can be so easily and efficiently implemented.

2. Experimenting with Collection

We constructed experimental collectors for p-groups, but many of our conclusions should hold for soluble groups.

Our main experimental collector allows the user to associate an integer b_{ij}, the *badness*, with each word w_{ij} of (1), and so with every minimal non-normal word. The user also specifies a choice algorithm, e.g. choose the leftmost, choose the rightmost, choose at random. Suppose now we are collecting the word w. The basic collection step of the experimental collector is to apply the choice algorithm to the minimal non-normal subwords of w whose associated badness is minimal, and then to collect the chosen subword. For example, the experimental collector performs collection to the left when $b_{ij} = i$ and the choice algorithm is to choose the leftmost; collection from the right is done when $b_{ij} = constant$ and the choice algorithm is to choose the rightmost. Of course not all collection strategies can be described in this way.

We have also programmed an experimental collector for *top-down collection*, detailed in Fig. 2. The aim of top-down collection is to bring together generators a_i through normal words so that a power substitution $a_i^{p_i} = w_{ii}$ can be made. We say that (an occurrence of) a_i is *uncollected* if it is involved in a minimal non-normal subword of w.

(1) If w is a normal word then stop.
(2) Set $i := \min\{j | \text{there exists an uncollected } a_j\}$, and set r to be the rightmost uncollected occurrence of a_i.
(3) If r is the only uncollected occurrence of a_i, set l to be the first letter of w, else set l to be the letter in w immediately following the second from the right uncollected occurrence of a_i.
(4) Set u to be the subword of w starting with l and ending just before r, and recursively apply this algorithm to transform u into a normal word.
(5) If r is no longer uncollected then goto step (1). If r is involved in a subword $m = a_i^{p_i}$ then collect m and goto step (1).
(6) Now r is involved in a subword $m = a_j a_i$ $(i < j)$. Collect this subword, set r to be the occurrence of a_i in the $a_i w_{ij}$ replacing m, and goto step (5).

Fig. 2. Top-down collection of w.

We have performed many of our tests with groups $G(p, i)$, described below, certain p-groups of p-class (and class) i. These groups allow us to study the effect on collection of varying the prime and the class, and we remark that $G(p, i)$ has the largest possible derived length given its class. [We thank C. Murgatroyd for calculating consistent power commutator presentations for various $G(p, i)$.]

Let p be a prime and $\mathbf{Z}_p$ be the ring of p-adic integers. Define G_p to be the inverse image of a Sylow p-subgroup of $SL_4(p)$ under the natural homomorphism of $SL_4(\mathbf{Z}_p)$ onto $SL_4(p)$, and let $G(p, i)$ denote the quotient of G_p by $\lambda_{i+1}(G_p)$, the $(i+1)$-st term of the lower p-central series of G_p. (The lower p-central series of a group G is defined by

$$\lambda_1(G) = G, \quad \lambda_{i+1}(G) = [\lambda_i(G), G]\lambda_i(G)^p.)$$

If $p^{\nu(i)}$ is the order of $G(p, i)$ then

$$\nu(i) = 4, 8, 12, 15, 19, 23, 27, 30, \ldots \quad \text{for } i = 1, 2, 3, \ldots.$$

Table 1 compares ten strategies applied to collect the word

$$a_1^2 a_2^2 \cdots a_8^2 a_1^2 a_2^2 \cdots a_8^2$$

in $G(3, 8)$. "Powers have priority" means that the badness $b_{ii} = 1$, while $b_{ij} = 2$ when $i < j$. "Short words have priority" means that $b_{ii} = \text{length}(w_{ii}) - p$, and $b_{ij} = \text{length}(w_{ij}) - 1$ when $i < j$. The columns are to be interpreted as follows. The "powers" column gives the number of substitutions of the form $a_i^p = w_{ii}$ performed during the course of the collection. "Triv comms" gives the number of substitutions of the form $a_j a_i = a_i w_{ij}$ when $w_{ij} = a_j$, while "nontriv comms" gives the number of substitutions $a_j a_i = a_i w_{ij}$ when $w_{ij} \neq a_j$. "Max length" is the maximum length of the word being collected, taken over the entire collection process, and "total length" is the sum of the lengths of the words introduced during the collection process (this is taken to be the measure of the complexity of a collection and is precisely defined in Section 4).

Table 2 shows the effect of varying the class i when collecting

$$a_1^4 a_2^4 \cdots a_8^4 a_1^4 a_2^4 \cdots a_8^4$$

in $G(5, i)$, using collection from the left, collection from the right, and top-down collection.

We have done some experimentation with groups other than $G(p, i)$, and collection from the left is seen to outperform collection from the right by a factor that increases rapidly with the class. Collection from the right sometimes wins when very easy collections are

being performed, or when words with a bias to the right are collected. For example, Table 3 shows the statistics arising from the collection of

$$w = a_1 a_3 a_5 a_7 a_2 a_4 a_6 a_8 a_1 a_2 a_4 a_8 a_1 a_3 a_9 a_{27}$$

in the three-generator exponent-4 Burnside group $B(3,4)$ (of order 2^{69}), where collection from the right did better than collection from the left. Observe the two rightmost occurrences of a_1 in w. This pair will eventually coalesce (as $p = 2$), and happen, when w is collected from the right, to coalesce before they have done much damage. Top-down collection is designed to take advantage of such situations, and hence wins comfortably. Note that Vaughan-Lee (1990) demonstrates the definite superiority of collection from the left over that from the right in the Canberra nilpotent quotient program when calculating quotients of Burnside groups.

3. On Implementations of Collection from the Left

We have implemented collection from the left for soluble groups in the programming language C. Initial experiments with our soluble group collector suggest that the superiority of collection from the left over that from the right holds in general soluble groups as well as p-groups. Felsch has programmed a FORTRAN version of our C

Table 1. Collecting $(a_1^2 a_2^2 \cdots a_8^2)^2$ in $G(3,8)$

Powers	Triv comms	Nontriv comms	Max length	Total length
		collection from the left		
360	6469	482	137	1088
		choose leftmost, powers have priority		
360	6461	482	137	1088
		choose leftmost, short words have priority		
1588	87856	2837	77	4772
		collection from the right		
4466	233434	7200	73	13406
		choose rightmost, powers have priority		
4466	233435	7200	73	13406
		choose rightmost, short words have priority		
3779	204702	7219	89	11345
		choose at random		
1093	80698	1630	453	3287
		choose at random, powers have priority		
1008	62386	1529	344	3032
		choose at random, short words have priority		
1679	93721	2894	78	5045
		top-down collection		
372	6729	540	132	1124

collector for use in the SOGOS system (Laue, Neubüser & Schoenwaelder, 1984). Felsch (private communication) finds that for multiplying random words in large soluble groups, our from the left collector is much faster than his "AG-code" collector (Felsch, 1976), which is based on collection from the right and requires a preprocessing stage. However, in many moderately sized practical computations AG-code can be faster than our collector. This appears to be the case because it is not random words being multiplied in many soluble group algorithms, and AG-code, with its preprocessing, has the property that trivial commutations (i.e. when $w_{ij} = a_j$) create no work at all in the actual collection.

Collection from the left for p-groups, for use in the Canberra nilpotent quotient program (Havas & Newman, 1980), has been implemented by Vaughan-Lee (1988), and his algorithm incorporates combinatorial collection (see Havas & Nicholson, 1976). Vaughan-Lee finds that collection from the left yields improvements of factors of about two to more than ten over the Havas–Nicholson collector when calculating quotients of certain Burnside groups.

Vaughan-Lee has given a detailed description of his implementation of collection from the left, so we will just mention some of the ways that our collector, which is for the general soluble case, differs from his. First, given (1), we store conjugates w_{ij} instead of commutators c_{ij}, and we do not handle words involving only one generator in a special way. Since we cannot use weights for the general soluble case we do a very small amount of preprocessing to determine, for each i, the least $j > i$ such that $\langle a_i, \ldots, a_n \rangle$ centralises $\langle a_j, \ldots, a_n \rangle$. We do no form of combinatorial collection. Finally, we use a fourth stack in

Table 2. Collecting $(a_1^4 a_2^4 \cdots a_8^4)^2$ in $G(5, i)$

i	Powers	Triv comms	Nontriv comms	Max length	Total length
		collection from the left			
4	123	3503	416	108	596
5	320	10454	709	150	1591
6	768	24979	1458	206	3840
7	1518	48580	2770	290	7604
8	2531	88191	3390	356	12677
9	4761	187894	6613	425	23833
10	9045	411757	12444	531	45262
11	14692	754293	19429	668	73507
12	18719	1109947	23633	735	93648
		collection from the right			
4	631	66779	1440	84	3136
5	5977	773568	14880	103	29876
6	38943	6465375	64162	121	194715
7	183602	33423605	332821	136	918024
8	450799	86813781	920895	150	2254017
		top-down collection			
4	131	4013	488	109	636
5	403	11492	965	152	2006
6	916	25028	1852	212	4580
7	1687	51233	3097	303	8449
8	2612	84913	3745	372	13082
9	5429	182681	8035	446	27173
10	9924	379690	13548	568	49657
11	16296	676659	22167	785	81527
12	20421	951501	25210	831	102158

the implementation in order that the maximum stack depth is bounded by $n(n+1)/2$ when multiplying two normal words; otherwise the stack depth may depend on the exponents p_i, as well as n.

4. Complexity Analysis of Collection from the Left for p-groups

Our complexity analysis shows, roughly speaking, that collection from the left is polynomial in the class, while collection from the right is exponential. In fact this statement slightly flatters our result.

As a measure of the complexity of a collection, we take the sum of the exponent sums of the words introduced. Thus if $[a_j, a_i] = c_{ij}$ and $a_i^p = w_{ii}$, the cost of interchanging a_i and a_j, or of replacing a_i^p by w_{ii}, will be taken to be the exponent sum of c_{ij} or of w_{ii}, respectively.

Consider first an example. Clearly, the complexity of collection in an abelian group is independent of the strategy used. Let $G = \langle a_1, \ldots, a_n \rangle$ be the cyclic group of order p^n, with $a_i^p = a_{i+1}^{u_{i,i+1}} \cdots a_n^{u_{i,n}}$, where the $u_{i,j}$ lie in the range $0, \ldots, p-1$, with $u_{i,i+1} \neq 0$. Consider the cost of multiplying $a_1^{x_1} \cdots a_n^{x_n}$ by $a_1^{y_1} \cdots a_n^{y_n}$. If $x_1 + y_1 \geq p$ this is the cost of multiplying $a_2^{x_2} \cdots a_n^{x_n}$ by $a_2^{y_2} \cdots a_n^{y_n}$, and then multiplying this word by $a_2^{u_{12}} \cdots a_n^{u_{1n}}$. If this overflow occurs at each step the total cost will be proportional to 2^n. If the $u_{i,j}$, and the x_i and y_i, are chosen at random, the probability of overflow at each step will be about $\frac{1}{2}$, at least after the

Table 3. Collecting $a_1a_3a_5a_7a_2a_4a_6a_8a_1a_2a_4a_8a_1a_3a_9a_{27}$ in $B(3,4)$

Powers	Triv comms	Nontriv comms	Max length	Total length
	collection from the left			
386	6895	305	179	783
	choose leftmost, powers have priority			
386	6895	305	179	783
	choose leftmost, short words have priority			
310	10761	343	52	631
	collection from the right			
216	7675	207	52	443
	choose rightmost, powers have priority			
216	7675	207	52	443
	choose rightmost, short words have priority			
168	5874	182	55	347
	choose at random			
764	32445	820	264	1539
	choose at random, powers have priority			
878	42596	951	313	1767
	choose at random, short words have priority			
308	12014	353	55	627
	top-down collection			
97	1835	87	104	205

first step, since $u_{i,i+1} \neq 0$. This gives an expected cost proportional to $(\frac{3}{2})^n$. To prove that a method of collection is polynomial in the class we must therefore get round this example. One method is to take a definition of class which makes the class of a cyclic group an exponential function of its composition length. Another is to restrict ourselves to "efficient" presentations. We discuss both ideas, with emphasis on the former.

Let G be a p-group, and

$$G = \theta_1(G) \supseteq \theta_2(G) \supseteq \cdots \supseteq \theta_c(G) \supset \theta_{c+1}(G) = \langle 1 \rangle \tag{2}$$

be a chain of normal subgroups of G satisfying, for all $i, j > 0$,

(i) $[\theta_i(G), \theta_j(G)] \subseteq \theta_{i+j}(G)$,
(ii) $\theta_i(G)^p \subseteq \theta_{2i}(G)$,

where $\theta_k(G) = \langle 1 \rangle$ if $k > c$. Then (2) is called a *θ-central series* for G. Since repetitions are allowed, any central series can be refined by a θ-central series.

We shall now assume that our power commutator presentation for G is consistent, and that the central series defined by this presentation is refined by a fixed θ-central series (2) whose length c will be called the *θ-class* of G. If $g \in \theta_i(G) - \theta_{i+1}(G)$ then i will be called the *θ-weight* of g. The maximum number of pc-generators of the same θ-weight in a normal word, i.e. $(p-1) \times \max_i \dim_{\mathbf{F}_p}(\theta_i(G)/\theta_{i+1}(G))$, will be called the *θ-width* of G. Strictly speaking, of course, these are invariants of the series (2). Clearly, condition (ii) forces the θ-class of a cyclic group to be exponential in the composition length.

For fixed $c > 0$ and $w > 0$ define the functions $\alpha(x, y)$ and $\beta(x)$ for positive integers x and y with $x \geq y$ by

$$\alpha(x, y) = 0 \quad \text{if } x + y > c,$$

else

$$\begin{aligned} \alpha(x, y) &= w^2 \sum_{t=x+y}^{c} (c - t + 1) + w \sum_{z=x}^{x+y} \alpha(x+y, z) + w \sum_{z=x+y+1}^{\lfloor c/2 \rfloor} \alpha(z, z) + w \sum_{z=x}^{\lfloor c/2 - y \rfloor} \beta(y+z). \\ \beta(x) &= w \sum_{z=x}^{\lfloor c/2 \rfloor} \alpha(z, z). \end{aligned} \tag{3}$$

We shall always take c to be the θ-class and w to be the θ-width of some fixed p-group G. Our main complexity results follow from the following lemma.

LEMMA 1.

(*a*) *The cost of collecting (from the left) ga_j, where $g \in G$ is a word in normal form of θ-weight x, and a_j is a pc-generator of θ-weight $y \leq x$, is at most $\alpha(x, y)$.*

(*b*) *The cost of collecting (from the left) the product gh of two elements of G of θ-weight x in normal form is at most $\beta(x)$.*

PROOF. If $x + y > c$ then $\alpha(x, y) = 0$, and if $2x > c$ then $\beta(x) = 0$, so the results hold in these cases.

Suppose now that $x + y = k \leq c$, and that (a) holds for $x + y > k$. Then (b) holds for $2x > k$. Consider collecting ga_j where g is of θ-weight x, and a_j is of θ-weight y, where $g = b_1 \cdots b_m$ in normal form, so each 'b' is an 'a'. If $b_1 = a_i$ then we may assume that either $i > j$ or $i = j$ and $b_1 = \cdots = b_{p-1}$.

Case 1. $i > j$.

The first step is to write

$$b_1 \cdots b_m a_j = a_j b_1 [b_1, a_j] b_2 [b_2, a_j] \cdots b_m [b_m, a_j]$$

at a cost of at most $w^2 \sum_{t=x+y}^{c} (c-t+1)$. Then, one must postmultiply a normal word in turn by $b_2, [b_2, a_j], b_3, \ldots$. If a generator b_t is of θ-weight z, the cost of postmultiplying by b_t will be at most $\alpha(x+y, z)$ if $z \leq x+y$, and at most $\alpha(z, z)$ otherwise. The cost of postmultiplying by $[b_t, a_j]$ will be at most $\beta(y+z)$, so the total cost is at most the value given by (3), as required.

Case 2. $i = j$, $b_1 = \cdots = b_{p-1}$.

The first step is to write

$$b_1 \cdots b_m a_j = a_j^p b_p [b_p, a_j] \cdots b_m [b_m, a_j]$$

and the calculation proceeds as before, with a slightly smaller bound. It is at this point that we use the fact that $\theta_i(G)^p \subseteq \theta_{2i}(G)$.

For our next result we need the Fibonacci numbers, defined by $F_0 = F_1 = 1$, and $F_n = F_{n-1} + F_{n-2}$ if $n \geq 2$.

LEMMA 2.

(a) *Let $i = i_{xy}$ be the least integer such that $xF_i + yF_{i-1} > c$, where $x \geq y > 0$ and $x + y \leq c$. Then $\alpha(x, y)$ is a polynomial in w with non-negative coefficients of degree at most i.*

(b) *If j is the least integer such that $xF_j > c$, then $\beta(x)$ is a polynomial in w with non-negative coefficients of degree at most j.*

PROOF. Let $i_{xy} = k$, where $k \leq c$, and assume that (a) holds when $i_{xy} < k$ and that (b) holds when $i_{xx} < k$. If $k = 2$ then $2x + y > c$ so

$$\alpha(x, y) = \binom{c-x-y+2}{2} w^2,$$

as required. Also, if $i_{xx} = 2$ then $\beta(x)$ is of degree 3 in w, and if j is defined as in (b) then $j = 3$. In general, (3) gives $\alpha(x, y)$ as a sum of four polynomials of degrees at most $2, k, k$ and k, so $\alpha(x, y)$ is of degree at most k in w, with non-negative coefficients. Similarly, (b) holds as well.

It is now easy to obtain our main result on the complexity of collection from the left.

THEOREM 3. *The cost of multiplying two words in normal form by collection from the left is bounded above by a polynomial in w of degree less than $2 + \log_\kappa c$, where κ is the golden ratio $(1+\sqrt{5})/2$.*

PROOF. It is easy to see, and well known, that $\kappa^{n-1} < F_n < \kappa^n$ for $n > 1$. Now we need to calculate the degree in w of $\beta(1)$, which is j, where j is the least integer such that $F_j > c$. So $\kappa^{j-2} \leq F_{j-1} \leq c$. This gives $j \leq 2 + \log_\kappa c$, as required.

A similar argument gives the following more precise result.

THEOREM 4.

$$\alpha(x, y) = o((w(c-x-y+1)/2)^{\log_\kappa(c/2(\kappa x+y))}w^2),$$

$$\beta(x) = o((w(c-2x+1)/2)^{\log_\kappa(c/2x)}w^2).$$

Calculation shows that in fact $\log_\kappa \beta(1) \simeq 0{\cdot}8 \log_\kappa(wc/2)\log_\kappa(c)$ for small values of w and c, so it appears that, in the above estimates, logarithms can be taken to the base $\kappa^{5/4} \simeq 1{\cdot}825$ rather than to the base κ. In practice, of course, these bounds are rather high, as we have assumed the worst possible presentation. It is tempting, but false, to suppose that one could replace w by the average number of generators of the appropriate θ-weights (counting exponents) that occur in the relations. Suppose that this average is 1. When the product of two words is collected, the collected part will build up to a "random" word, but where, in our analysis, we were considering having to take w generators of θ-weight i, say, through w generators of θ-weight j, each exchange adding w generators of θ-weight k for each $k \geq i+j$, we now consider having to take one generator of θ-weight i through w generators of θ-weight j, each exchange adding one generator of θ-weight k as before. Thus, in this case one would replace w by $w^{1/3}$. In any case, the above formulae show that collection from the left is much better than exponential in the class, and may be of some value in estimating the complexity of p-group algorithms.

We now discuss the possibility of obtaining a similar complexity analysis when the p-class, or even the nilpotency class, is used instead of the θ-class. We have seen that this requires us to use a pc-presentation with favourable pth powers. Suppose then that we replace condition (ii) on the series $\{\theta_i(G)\}$ by the condition that, for all i, the collected form of a_i^p involves at most e generators, counting exponents, of θ-weight less than twice the θ-weight of a_i. Then the above bounds for the complexity of collection from the left still hold with w replaced by $w(p+e-1)/(p-1)$, as a glance at the proof of Lemma 1, Case 2 shows. It seems that with many p-groups one can hope for a pc-presentation with $\theta_i(G)$ the ith term of the lower p-central series of G, and $e = 1$.

5. Comparing Collection from the Left with Collection from the Right

While collection from the left seems in general to be superior to collection from the right, there are situations where collection from the right wins. For example, if G has a pc-presentation on $a_1, \ldots, a_n$ with a_1 of order p, and $[a_2, a_1] \neq 1$, then collecting $a_2 a_1^p$ is obviously best done from the right. Of course, the word being collected here is not the concatenation of two normal words, and collection from the right is similarly embarrassed by $a_2^p a_1$. Having admitted that collection from the left is not always superior, we now analyse the performance of the two algorithms in collecting the product of two normal words in the wreath product $G = C_p \wr C_p$.

It is easy to see that G has a pc-presentation with generators $a_1, \ldots, a_{p+1}$ of order p, where $[a_i, a_1] = a_{i+1}$ if $i > 1$ (we define $a_i = 1$ if $i > p+1$), and $[a_i, a_j] = 1$ if $i > j > 1$. It is hard to imagine a pc-presentation of a group of class p with a more favourable presentation for efficient collection. Let $f(i, j)$ denote the cost of collecting $a_i a_1^j$ from the right. Since $a_i a_1^j = a_1 a_i a_{i+1} a_1^{j-1}$, one gets

$$f(i, j) = f(i, j-1) + f(i+1, j-1) + 1$$

if $1 < i \leq p$, and $f(i, 0) = f(p+1, j) = 0$. This gives $f(i, j) = 2^j - 1$, provided that $i+j \leq p+1$. So the cost of collecting $a_2 a_1^{p-1}$ from the right is $2^{p-1} - 1$, already exponential in the class.

The only possible improvement in the presentation would arise from swapping a_1 and a_2. But this makes hardly any difference; for example, consider collecting $a_3 a_2^{p-1}$ with this new presentation. Now consider the effect of taking a more malicious presentation of G, so that the relations become $[a_i, a_1] = a_{i+1}^{p-1}$, for $i > 1$, pth powers and other commutators being still trivial. This raises the complexity of collecting $a_2 a_1^{p-1}$ to $p^{p-1}-1$. As we can take $c = p+1$ and $w = p-1$, in the notation of the last section, we see that collection in G is worse from the right in the most favourable circumstances than is collection from the left in any presentation whatever with these values of w and c (except perhaps for small p), and furthermore, collection from the right in G is very sensitive to hostile presentations.

Now consider collection from the left in this group. With the friendly (first) presentation, collecting va_1, where v is any normal word, costs at most $(p-1)^2$ (one for each noncentral generator that a_1 must cross), so collecting $a_2 a_1^{p-1}$, or even va_1^{p-1}, costs at most $(p-1)^3$. Thus, the cost of collecting the product of any two normal words from the left is at most $(p-1)^3$. If we now take the unfriendly presentation of G, this only multiplies these estimates by $p-1$, and if we take the worst possible presentation, subject to a_2 lying in the maximal abelian subgroup, it is still only $O(p^5)$. This can be compared to the bound of Theorem 4, which in this case is $O((p^3/2)^{\log_4((p+1)/2)})$; in this group the log in the exponent does not arise since we have a maximal elementary abelian subgroup.

In a more general vein, we can estimate the complexity of collecting from the right the concatenation of two words in normal form in an arbitrary pc-presentation, as we did for collection from the left in the previous section. We end up, in the worst case, but ignoring powers, with a bound that is a polynomial in w of degree $2c-1$ in w, with leading term $w^{2c-1}/c!$ This lower bound is only valid if we ignore the fact that presentations are supposed to be consistent (or consistent modulo their centres, as in the NQA). Nonetheless, the cost of collecting $a_2 a_1^{p-1}$ in our hostile presentation of G, which we can regard as having $w = p-1$ and $c = p+1$, is not so far off this limit, and we could have taken a nastier presentation still.

6. Conclusion

Collection from the left is superior to collection from the right in general, and in the "worst case" they are of complexity approximately $(wc)^{\log c}$ and w^c, where w and c are defined in section 4, but in any group some words collect faster from the right. Top-down collection apparently makes the best of both worlds, but it is not clear whether it is as good, on average, as collection from the left, and it is unlikely that it can be implemented so efficiently.

If, as we expect, collection from the left is to become the standard, this will affect algorithm design. Note that collection from the left allows one to collect $u^{-1}v$ as fast as uw, where u, v and w are normal words, with $w = u^{-1}v$. The idea is to solve $ux = v$; write $u = a_1^{u_1} \cdots a_n^{u_n}$, and similarly for v and x. Then $x_1 = v_1 - u_1 \bmod p$, so one collects $a_1^{x_1}$ across u, then calculates x_2, etc. (It appears that the Felsch collector calculates inverses faster than it multiplies. The explanation is, presumably, that the inversion algorithm he uses, though different from the above, moves away from collection from the right.) Thus, calculating $u^{-1}vu$ is equivalent to two multiplications, and $u^{-1}v^{-1}uv$ is equivalent to three, whereas calculating uvu^{-1} and $uvu^{-1}v^{-1}$ are equivalent to three and four multiplications, respectively. Another issue is the order in which these multiplications are carried out. The natural order would seem to be $(u^{-1}(v^{-1}(uv)))$, i.e. from the right. For example, if u and v happen to commute, the last of these three operations is trivial. More generally, if $h(i,j)$ is

the average time taken to collect the product of two words of weights i and j [the weight of an element being i if it lies in the ith but not the $(i+1)$th term of the lower p-central series of G], then the cost of calculating the commutator in this way should, on average, be $h(j, i)+h(j, j)+h(k, k)$, where i, j and k are the weights of u, v and $[u, v]$, and this is easily seen to be better than the expected time when any other bracketing is used. But the unexpected can happen. If $u=v$ calculating $v^{-1}u$ involves no collection at all.

A more fundamental issue is the question of producing pc-presentations that allow efficient collection. It seems from our analysis that collection from the left particularly favours presentations that have friendly pth powers. The reason being that if a_i has weight i then a_i^p can have weight as little as $i+1$, whereas the commutator of two generators of weight i is at least $2i$. With collection from the right, on the other hand, when one may keep on dragging generators over generators of weight 1, it is presumably more important to keep commutators under control.

Finally, we believe that the superiority of collection from the left over collection from the right is even more marked in general soluble groups than in p-groups. The evidence for this is experimental and heuristic and we offer no analysis.

We thank M. F. Newman for carefully reading a draft of this paper and making helpful comments.

References

Felsch, V. (1976). A machine independent implementation of a collection algorithm for the multiplication of group elements. In *Proc. 1976 ACM Symp. on Symbolic and Algebraic Computation* (Jenks, R. D., ed.). New York: Assoc. Comput. Mach., pp. 159–166.

Hall, P. (1934). A contribution to the theory of groups of prime-power order. *Proc. London Math. Soc.* **36**, 29–95.

Havas, G., Newman, M. F. (1980). Applications of computers to questions like those of Burnside. *Lecture Notes in Mathematics* **806**, 211–230. Berlin: Springer.

Havas, G., Nicholson, T. (1976). Collection. In *Proc. 1976 ACM Symp. on Symbolic and Algebraic Computation* (Jenks, R. D., ed.). New York: Assoc. Comput. Mach., pp. 1–14.

Laue, R., Neubüser, J., Schoenwaelder, U. (1984). Algorithms for finite soluble groups and the SOGOS system. In *Computational Group Theory* (Atkinson, M. D., ed.). London: Academic Press, pp. 105–135.

Neubüser, J. (1961). Bestimmung der Untergruppenverbände endlicher p-Gruppen auf einer programmgesteuerten elektronischen Dualmaschine. *Numer. Math.* **3**, 271–278.

Sims, C. C. (1987). Verifying nilpotence. *J. Symbolic Comp.* **3**, 231–247.

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