BULL. AUSTRAL. MATH. SOC. 20F05, 20-04
VOL. 62 (2000) [105–118]

PROVING A GROUP TRIVIAL MADE EASY: A CASE STUDY IN COSET ENUMERATION

GEORGE HAVAS AND COLIN RAMSAY

Dedicated to B.H. Neumann, on the occasion of his 90th birthday

Coset enumeration, based on the methods described by Todd and Coxeter, is one of the basic tools for investigating finitely presented groups. The process is not well understood, and various pathological presentations of, for example, the trivial group have been suggested as challenge problems. Here we consider one such family of presentations proposed by B.H. Neumann. We show that the problems are much easier than they first appear, albeit at the expense of considerable preliminary 'experimentation'. This demonstrates how far the range of applicability of coset enumeration has improved.

1. INTRODUCTION

Coset enumeration, as a technique for the investigation of finitely presented groups, was systematised and popularised by Todd and Coxeter [**16**]. The earliest computer implementation seems to have been by Haselgrove, in 1953. This, along with other early implementations, is described by Leech [**9**]. Detailed accounts of the techniques used in coset enumeration can be found in [**1, 3, 10, 11, 14**].

Coset enumeration takes as input a finitely presented group and a finitely generated subgroup, and attempts to find the index of the subgroup in the whole group. In principle, it will succeed whenever this index is finite. However, the Todd-Coxeter process is not, in general, an algorithm, and its behaviour and computational complexity are not well-understood. Sims [**14**] has shown that there is no polynomial bound, in terms of the maximum number of cosets, for the number of coset tables which can be derived by coset table operations such as those used in enumeration programmes.

The space available during an enumeration determines the maximum number of cosets (MaxCos, or simply M) which can be 'active' at any one time during an enumeration. Sims' result means that the total number of cosets (TotCos, or simply T) defined

Received 15th October, 1999
This work was supported by the Australian Research Council. We are grateful to The University of Queensland's High Performance Computing Unit for access to the Origin 2000 supercomputer.

can be very much larger than MaxCos, and that MaxCos can be very much larger than the (finite) index. In general, the running time of the Todd-Coxeter process depends on T, while the memory requirements are bounded below by M.

In this paper we consider a presentation of the trivial group which has been proposed as a challenge to machine implementations of the Todd-Coxeter process. We discuss how the features of a modern coset enumeration programme can be used to investigate this presentation; it is very pathological, with $T \gg M \gg 1$, but is a good deal easier to prove trivial than first attempts suggest.

We adopt the convention of using lower-case letters for generators, with upper-case letters denoting inverses; for example, $ABab$ stands for $a^{-1}b^{-1}ab$. Consider the group

$$E_1 = \langle r, s, t \mid TrtRR, RsrSS, StsTT \rangle.$$

That E_1 is the trivial group is not obvious (see Higman [**6**], Neumann [**12**]). Neumann [**13**] notes that the presentation given for E_1 was very difficult for early computer implementations of the Todd-Coxeter process, although modern implementations solve it readily. He notes that another presentation of the trivial group can be obtained by replacing r, s and t in the presentation for E_1 by $TrtRR$, $RsrSS$ and $StsTT$ respectively. This gives the group

$$\begin{aligned} E_2 = \langle r, s, t \mid\ & ttSTsTrtRRStsTTrrTRtrrTRt, \\ & rrTRtRsrSSTrtRRssRSrssRSr, ssRSrStsTTRsrSSttSTsttSTs \rangle, \end{aligned}$$

which "beats all existing programmes on all existing computers". Neumann also notes that an infinite sequence of presentations for the trivial group, of ever-increasing difficulty, can be obtained by iterating this process.

Sims [**14**, Section 5.8] discusses these presentations, and notes that the coset enumeration procedures which he describes were unable to find any coincidences (that is, to prove any two distinct words equal) in E_2 in defining 100000 cosets. He goes on to discuss how the Knuth-Bendix process (originally described in [**8**]) is able to prove E_2 trivial using "well under a megabyte" of memory. We show that we can, in fact, prove E_2 trivial by coset enumeration, and that this proof is, in retrospect, 'easy'.

The coset enumeration implementation which we used was ACE (advanced coset enumerator) by Havas and Ramsay [**4**], which is an enhanced version of the one described in [**3**]. The enumerations were performed on an SGI Origin 2000 computer which was equipped with enough memory to allow the definition of some hundreds of millions of cosets.

2. Initial Investigations

Although coset enumeration for E_1 is easy, it is pathological. A standard Felsch (see below) enumeration over the trivial subgroup, which we denote by $E_1/\langle 1 \rangle$, yields

$T = 588$, see [**14**, p248]. Using our enumerator, and techniques similar to those to be described for E_2, the best machine enumeration for $E_1/\langle 1\rangle$ that we achieved had $M = 79$ and $T = 81$. Hand-pruning of the definition sequence reduced this to $M = T = 64$. Although not of any direct help in enumerating $E_2/\langle 1\rangle$, these results do suggest that we might be able to reduce the pathologicality of E_2 sufficiently to enable an enumeration to complete using a 'reasonable' amount of time and space.

The three key features to our approach for E_2 are: the availability of a machine with a large amount of memory; the choice of enumeration strategy; and the choice of presentation. Once we know that we can complete an enumeration in the space available, then we can attempt to reduce the MaxCos/TotCos values (or the running time). However, for the initial attempts, we have no upper bound on the space requirements, so we would like as much memory as possible. It will turn out that the enumeration for E_2 can be done in about 4 MByte, while a straightforward enumeration of E_2 as it stands requires about 8 GByte.

During an enumeration, we have considerable freedom of choice regarding the order in which we define cosets. Two standard techniques are to define new cosets using the next empty position in the coset table (traditionally called the Felsch strategy) or using the relator tables (traditionally known as the HLT strategy). In Felsch strategies each definition is tested against all essentially different positions in all the relators until all consequent deductions and coincidences have been found and processed. In HLT strategies each coset is scanned against all relators, with definitions made as necessary to close all scans, while processing any deductions or coincidences encountered.

To select our enumeration strategy, we undertook some preliminary investigations of $E_2/\langle 1\rangle$ where we put a limit on TotCos and noted how many cosets remained active when this limit was reached; that is, how many coincidences there had been. The results are summarised in Table 1. The figures for the number of active cosets remaining for a given choice of TotCos are given in the last four columns. The first column gives the strategy, with n/m indicating a mixed strategy in which alternate blocks of n Felsch definitions and m HLT coset scans were made, with all definitions tested against all essentially different positions. Note that m coset scans can result in up to $75m$ coset definitions, since the total length of the relators is 75.

These results confirm Sims' observations, and suggest that Felsch strategies are 'bad' and that HLT ones are 'good'. They also suggest that HLT strategy definitions should be tested against all essentially different positions as soon as possible after they are made. (In the mixed strategies, these definitions are tested after the block of m coset scans has been completed.) ACE has a mode whereby all the definitions are made using the HLT strategy, while definitions are tested against all essentially different positions after each individual coset/relator scan. Since our primary concern is to minimise the space requirements, this mixed strategy is the strategy we adopted; unless otherwise noted, it

Table 1: Progress in various strategies

	TotCos (millions)			
strategy	0.1	1	10	100
Felsch	100000	1000000	10000000	100000000
HLT	100000	999811	6787119	33847599
10000/1	100000	1000000	10000000	100000000
1000/1	100000	1000000	10000000	99999985
100/1	100000	999962	9999855	73563024
10/1	99988	999780	7711032	44379346
1/1	99976	999715	6368554	47870176
1/10	99976	999709	6372346	47678685
1/100	100000	999797	6786909	37167035
1/1000	99988	999815	6787109	33234182

Table 2: The initial runs

	E_2		E_2'	
subgroup	MaxCos	TotCos	MaxCos	TotCos
$\langle r\rangle$	50561877	122400832	50560999	122078184
$\langle s\rangle$	70169472	166476400	70168594	166907903
$\langle t\rangle$	43277069	102965293	69609426	170455932
$\langle r,s\rangle$	5408953	11368023	5351144	11308278
$\langle r,t\rangle$	5524264	11318575	5523386	11263377
$\langle s,t\rangle$	19013781	48598546	19012903	49144096

was used for all the enumerations reported herein.

We started with enumerations of E_2 over one- and two-generator subgroups. Our enumeration strategy is obviously sensitive to the order of the relators. However, if we test over all one- and two-generator subgroups, then we need test only two of the six possible orderings, since cycling the generators induces a cycling of the relators. All of the twelve possible combinations were tested, using a memory allocation sufficient for a table size of 200 million cosets, and they all completed successfully. The results are tabulated in Table 2, where E_2' is E_2 with the last two relators swapped. These results are the first successful coset enumerations to be performed in E_2 over 'sensible' subgroups.

3. Success

The results in Table 2 suggest that enumeration over the trivial subgroup would be possible, albeit in an uncomfortably large number of cosets. However, instead of attempting this immediately, we first try to reduce the totals for enumerations over the one-generator subgroups. Experience with other enumerations indicates that a 'good' enumeration over one subgroup often translates to a good enumeration over a smaller subgroup. Apart from reordering the relators, it is well-known that rotating and inverting relators can also have a significant effect on performance (see Cannon, Dimino, Havas and Watson [1] and Havas and Ramsay [5]). We would expect this to be particularly effective for E_2, given that the three relators in the presentation have many common subwords.

ACE has the ability to generate and use presentations which are equivalent to the given presentation, but where the relators have been randomly reordered, rotated, and inverted. We can run many tests using this feature, and select those which yield the smallest T and M values. To maximise throughput, we put a tight limit on the amount of space; enumerations which cannot complete in the given space tend to overflow quickly, and can be aborted.

For our tests we ran each of the three one-generator subgroup enumerations on 5000 randomly generated equivalent presentations, with a limit of 10 million on TotCos. There were a total of 23 successes in the 15000 runs, nine over $\langle r \rangle$, eight over $\langle s \rangle$ and six over $\langle t \rangle$. The ranges for T, M and T/M were 4171770–9946881, 496442–1119111 and 6.76–9.33 respectively. These 23 presentations all enumerated successfully over the trivial subgroup, with a limit of 100 million on TotCos. The ranges for T, M and T/M were 36249963–71143046, 4362123–6164681 and 7.48–11.54 respectively.

The presentations which achieved $T = 36249963$ (with $M = 4826625$) and $M = 4362123$ (with $T = 43608516$) were respectively

$$\begin{aligned} E_2^a = \langle r, s, t \mid & ttSTsssRSrStsTTRsrSSttSTs, \\ & ssRSrssRSrrrTRtRsrSSTrtRR, TrtRRttSTsrrTRtStsTTTrtRR \rangle, \\ E_2^b = \langle r, s, t \mid & ttSTsTrtRRStsTTrrTRtrrTRt, \\ & RsrSSttSTsttSTsssRSrStsTT, RsrSSrrTRtssRSrTrtRRRsrSS \rangle. \end{aligned}$$

Note how these tend to have five-letter subwords or their inverses (that is, the relators for E_1) repeated at the beginning and end of the relators. The other 21 presentations displayed a similar pattern. This pattern however seems to be 'delicate', and attempts to achieve lower M and T values by manually altering the presentation, using E_2^a and E_2^b as guides, met with no success.

For comparison purposes, we returned to the initial presentation E_2, and attempted the enumeration $E_2/\langle 1 \rangle$. Over a long weekend, when the machine we used was not busy, the enumerator was given enough memory for a table of 400 million cosets (that is, 8.94 GByte), with space recovery via table compaction enabled. This enumeration completed successfully, with $T = 825673759$ and $M = 343662508$, after 3 compaction phases (that is, table overflows). Note that the ratio $T/M = 2.40$ is in line with those from Table 2, which range from 2.04 to 2.58.

4. Interlude

The metrics normally used to measure the cost of an enumeration are the values of T and M and the running time. However, given that there may be many users competing for these resources, a better metric might be the memory-time integral. This can be approximated by the product of memory allocation and running time. One approach to minimising this cost is to parallelise (parts of) the enumeration, as discussed by Cooperman and Havas [2]. The large values of T/M for the presentations E_2^a and E_2^b suggests an alternative approach; that is, run the enumeration with a memory allocation somewhere between M and T, recovering the space occupied by redundant cosets by compacting the table each time it overflows.

A series of tests using E_2^b was run using this idea, and the results are given in Table 3. The total running time is split into that spent enumerating cosets and that spent compacting the table, with the fourth column recording the number of compaction phases. (The usual caveats regarding running times on multiuser systems apply.) The Origin 2000 machine has a multiprocessor shared-memory architecture. Although all the memory is shared between all processors, physically it is divided between the processors as local memory. This, coupled with the usual caching behaviour, explains the general reduction in enumeration time as the memory allocation decreases. Note also that the time to compact the table is not fixed; it varies between runs due to the variation in table size, and it varies within a run as the proportion of the table containing active cosets varies. All these effects, to some extent, cancel each other out. So the total cost of the enumeration scales linearly with the memory allocation, except towards the smaller allocations where the enumeration time is static and the compaction cost increases rapidly.

5. Smaller Success

The large values of T/M observed in our best enumerations imply that there are many coincidences before the coincidence which precipitated the final collapse to an index of one. If the progress of the enumeration is monitored, it is seen that coincidences and partial collapses occur cyclically; the number of active cosets builds up steadily, with very few coincidences, then there is a coincidence which deletes a significent number of cosets, and then the cycle repeats. The cycles are not identical, but they are very similar, suggesting that the enumerator is repeatedly proving the same relation (or, at least, variants of the same relation) and discarding it after 'using' it. (Contrast this with the Knuth-Bendix process, which does keep a 'record' of previous results.)

All 23 of our successful presentations displayed a similar pattern of activity, and we chose to investigate E_2^a in more detail. There are 246 cycles in the enumeration for E_2^a. Of these, 231 are similar (the 'standard' cycle, described below), and the remaining fifteen are non-standard (eight are shorter than standard and seven are longer). If we

Table 3: The compacting runs

memory	time (seconds)		#comp	memory-time	
(10^6 cosets)	enum	comp		(10^9 byte-seconds)	norm
45	442.30	0.00	0	477.68400	1.000
40	439.30	6.69	1	428.15040	0.896
35	438.14	5.97	1	373.05240	0.781
30	436.81	5.03	1	318.12480	0.666
25	436.72	4.16	1	264.52800	0.554
20	434.76	7.97	2	212.51040	0.445
15	432.71	10.28	3	159.47640	0.334
10	430.81	13.98	5	106.74960	0.223
9	430.37	16.48	6	96.51960	0.202
8	430.57	18.29	7	86.18112	0.180
7	429.15	22.94	9	75.95112	0.159
6	428.56	30.05	12	66.03984	0.138
5	428.58	51.26	20	57.58080	0.121
4.5	428.73	85.81	32	55.54440	0.116

ignore the final collapse to index one and the coincidences which occur during a cycle, then the average total number of cosets defined per cycle is $36249963/246 = 147358$, the average increase in the number of active cosets per cycle is $4826625/246 = 19620$, and the average size of the partial collapse at the end of a cycle is $147358 - 19620 = 127738$.

In a standard cycle the words involved in the fourteen (primary) coincidences can be partitioned into heads and tails. The (ordered) set of tails is the same from cycle to cycle, while the heads change each cycle, but are fixed for the duration of the cycle (there are always two distinct heads). (This has not been formally checked for all 231 cycles, but has been verified 'manually'.) Table 4 illustrates this, where a, b, c and A, B, C have been used for the words $TrtRR$, $RsrSS$, $StsTT$ and their inverses in the tails.

This repeated proving and discarding of the 'same' relations suggests that the performance of the enumerator could be enhanced by adding the relation which triggers the first partial collapse to the relators and restarting the enumeration. If coincidence #14 from cycle #1 is added to the presentation, then the resulting enumeration completes in 26.13 seconds, with $M = 125332$ and $T = 1524984$. (This new enumeration displays cyclic behaviour similar to E_2^a, but the cycles are shorter.) If we repeat this procedure twice more, we arrive at a six-relator presentation which enumerates in 0.25 seconds, with $M = 1656$ and $T = 13331$. The second and third added coincidences are, respectively, $cc = ABcBAA$ and $cBttST = cBCCBS$; the last of these reduces to the relator bc. In fact, the four-relator presentation $bc + E_2^a$ enumerates in 0.12 seconds with the same T and M statistics, while $E_2^a + bc$ enumerates in 0.11 seconds with $M = 1501$ and $T = 12765$.

Table 4: The standard cycle

cycle	coinc	word 1		word 2	
1	1	$rrTRt$	c	$StsTT$	AA
	2	$rrTRt$	$ABaabC$	$StsTT$	$BCbACaccB$
	...				
	13	$rrTRt$	$cBaa$	$rrTRt$	$CaBBc$
	14	$rrTRt$	$BabCAB$	$StsTT$	$BAABC$
2	1	$SrrTRt$	c	$SStsTT$	AA
	2	$SrrTRt$	$ABaabC$	$SStsTT$	$BCbACaccB$
	...				
	13	$SrrTRt$	$cBaa$	$SrrTRt$	$CaBBc$
	14	$SrrTRt$	$BabCAB$	$SStsTT$	$BAABC$
...					
246	1	$StssssssTRt$	c	$StsRRtsTT$	AA
	2	$StssssssTRt$	$ABaabC$	$StsRRtsTT$	$BCbACaccB$
	...				
	13	$StssssssTRt$	$cBaa$	$StssssssTRt$	$CaBBc$
	14	$StssssssTRt$	$BabCAB$	$StsRRtsTT$	$BAABC$

So we can prove E_2^a trivial by running three enumerations to their first partial collapse, adding the collapse-inducing coincidence to the relators each time, and then running the final enumeration to its conclusion. The four enumerations take a total of 2 seconds, and the memory requirement is determined by the space needed to complete the first cycle in the first enumeration; that is, $M = 163564$ and $T = 163839$. A quick test of the other 22 presentations revealed that the best values for memory usage (to the first partial collapse) were $M = 162855$ and $T = 162867$, so the memory requirement could be marginally reduced if required.

6. Faster Success

Both our mixed enumeration strategy and the HLT strategy make all definitions via coset/relator scans. In our mixed strategy, definitions are tested immediately in all essentially different positions; this ensures that the total number of cosets defined is minimised, but it is expensive computationally. In the HLT strategy, definition testing is, in effect, deferred until it occurs 'naturally' during coset/relator scans; this increases the total number of cosets defined, but processes cosets faster.

To compare the two approaches we revisited the four presentations E_2, E_2', E_2^a and E_2^b and the seven subgroups $\langle r,s\rangle$, $\langle r,t\rangle$, $\langle s,t\rangle$, $\langle r\rangle$, $\langle s\rangle$, $\langle t\rangle$ and $\langle 1\rangle$. Each of the 28 possible combinations was attempted for both strategies. Once again, enough memory for a table of 400 million cosets was allocated and space recovery enabled. The results are given in Table 5. The CPU times are in seconds, with the number of compactions (if any) in brackets afterwards. Note that the $E_2/\langle 1\rangle$, $E_2'/\langle 1\rangle$ and $E_2^b/\langle 1\rangle$ enumerations using HLT strategy are the first reported successful enumerations to define a total of more than 10^9 (or 2^{30}) cosets.

Dividing the running time by the total number of cosets defined gives a measure of a strategy's throughput. If we ignore the enumerations over the trivial subgroup, where the results are distorted by the compactions, our mixed strategy has a throughput of 10.40–12.14 seconds per million total cosets, while the HLT strategy achieves 3.62–4.32. The ratio between the two throughputs is 2.5–3.2. The HLT strategy is 'faster', however it defines more cosets. For the E_2 and E_2' presentations, the TotCos counts for the HLT strategy exceed those of our mixed strategy by a ratio of 1.63–2.00. The nett effect of these two factors is that the HLT strategy has a shorter running time for E_2 and E_2', even for the trivial subgroup enumerations.

Note that E_2^a and E_2^b were specifically selected for their good performance using our mixed strategy, and that they perform badly, in terms of the total number of cosets defined, using the HLT strategy. It is likely that the random presentation technique discussed in Section 3 would produce presentations that perform well using the HLT strategy, but this has not been checked. Finally, note that the throughputs (and their ratio) are specific to the type of presentation we are considering; three relators, each of length 25, and with each of the three generators equally represented. For other presentations, very different figures would be obtained.

7. Other Presentations

The construction method for the presentation E_2 suggests an alternative presentation for the trivial group, namely the six-generator presentation E_2^* given by

$$\langle r, s, t, x, y, z \mid ZxzXX, XyxYY, YzyZZ, x = TrtRR, y = RsrSS, z = StsTT\rangle.$$

ACE has no difficulty with $E_2^*/\langle 1\rangle$; using the Felsch strategy, with the gap-filling technique discussed in [3], it completes in 1.02 seconds, with $M = 54236$ and $T = 145725$. Some experimentation with the random equivalent presentation feature of ACE, and various mixed strategies, readily produced a strategy and presentation which completed in 0.29 seconds, with $M = 20619$ and $T = 47514$.

When we come to E_3, the next in our sequence of presentations for the trivial group, we have a choice of three-, six- and nine-generator presentations. We tested all of these presentations over $\langle 1\rangle$, with various strategies, and although some progress was made in some combinations, none looked particularly promising. However, we can use our success with E_2^a to bootstrap ourselves. Consider the presentations formed by adding the generators $\{x, y, z\}$ and the relations $\langle r = ZxzXX, s = XyxYY, t = YzyZZ\rangle$ to the

Table 5: Strategy Comparisons

strat	s/grp	count	E_2	E_2'	E_2^a	E_2^b
mixed	$\langle r, s\rangle$	M	5408953	5351144	382271	217923
		T	11368023	11308278	2404423	732645
		CPU	135.12	133.33	26.46	7.74
mixed	$\langle r, t\rangle$	M	5524264	5523386	189175	214583
		T	11318575	11263377	596049	807702
		CPU	137.45	135.37	6.36	8.40
mixed	$\langle s, t\rangle$	M	19013781	19012903	192995	387532
		T	48598546	49144096	603683	2894907
		CPU	568.76	574.58	6.47	30.59
mixed	$\langle r\rangle$	M	50561877	50560999	1075739	760500
		T	122400832	122078184	7484929	6546070
		CPU	1472.20	1465.59	83.34	73.18
mixed	$\langle s\rangle$	M	70169472	70168594	1056737	1021628
		T	166476400	166907903	7503652	9058194
		CPU	1957.70	1990.37	83.61	101.19
mixed	$\langle t\rangle$	M	43277069	69609426	808895	664835
		T	102965293	170455932	5438751	5468032
		CPU	1203.94	2026.93	60.74	60.96
mixed	$\langle 1\rangle$	M	343662508	343661630	4826625	4362123
		T	825673759	827238032	36249963	43608516
		CPU	10767.74 (3)	10661.63 (3)	398.11	477.66
HLT	$\langle r, s\rangle$	M	7616099	7614904	3850467	7456970
		T	19599655	19586109	5754224	20013546
		CPU	75.13	80.56	21.46	84.58
HLT	$\langle r, t\rangle$	M	9272920	9271725	1758680	7626160
		T	22581459	22522958	2410643	19070196
		CPU	85.47	92.29	8.80	79.74
HLT	$\langle s, t\rangle$	M	21686180	21951535	4324919	16981462
		T	84727676	85254987	6241018	78835721
		CPU	318.45	347.44	22.60	327.96
HLT	$\langle r\rangle$	M	54279676	54278481	17924337	38592075
		T	203304585	203012914	27578664	183544090
		CPU	865.03	847.40	110.01	792.19
HLT	$\langle s\rangle$	M	73663677	73662482	55394364	53829241
		T	277624534	278048035	90452331	256725138
		CPU	1194.01	1138.28	355.52	1093.57
HLT	$\langle t\rangle$	M	45954168	72291370	41527832	33913074
		T	168974036	277318454	65887654	154334228
		CPU	700.10	1149.52	255.01	648.95
HLT	$\langle 1\rangle$	M	346398343	346397148	277197842	246125047
		T	1346459378	1347224248	441951487	1239489022
		CPU	8062.55 (7)	7814.78 (7)	1973.22 (1)	6110.24 (4)

presentation E_2^a. We take this in two forms; E_3^a, where the new relations are added before the old ones, and E_3^b, where they are added after.

Using our mixed strategy, neither E_3^a nor E_3^b complete over $\langle 1\rangle$, but they both displayed the same cyclic behaviour as E_2^a. The cycles were somewhat longer than those of E_2^a, with those of E_3^b being shorter than those of E_3^a. Further, after defining a total of about 427 million cosets, there was a large collapse in E_3^b which reduced the number of active cosets from about 167 million to about 35 million. The cyclic behaviour then resumed.

This suggests that the enumeration would eventually complete, after some number of the larger cycles, and that the small and large cycles are associated, respectively, with the relators of E_1 and E_2, which are contained as subwords in the presentations E_3^a and E_3^b. If this is the case, and the pattern of nested cycles continues to grow for the sequence of presentations $\{E_n\}$, then we have a nice set of examples illustrating Sims' result.

The similarity of the cycles in the enumerations of E_3^a and E_3^b to those in E_2^a, and the fact that each of the new relators contain only one of the generators $\{r, s, t\}$, suggest that the coset definitions for the two sets of three relators are 'independent' of each other, at least in the early stages of an enumeration. This, in fact, seems to be the case, and if the techniques discussed in Section 5 are applied to E_3^a and E_3^b, then precisely the same sets of coincidences is obtained. (Again, this has only been checked manually.) So three partial enumerations prove that bc can be added to the relators of E_3^a and E_3^b. If this is done, then the first two coincidences in a subsequent enumeration prove that B and c (that is, $ssRSr$ and $StsTT$) can be added as relators.

If we now 'tidy-up' the presentation by taking, say, E_3^b, adding the relators $ssRSr$ and $StsTT$, and ordering the eight relators in length-increasing order, we obtain the presentation E_3^*. Since the triviality of a (that is, $TrtRR$) follows immediately from E_3^*, this presentation is, in some sense, an analogue of E_2^*. Using our mixed strategy, the enumeration $E_3^*/\langle 1\rangle$ completes in 325.41 seconds, with $M = 12020719$ and $T = 33740210$.

Of course, our 'success' with E_3^* is a trifle contrived, since it involved considerable non-determinism. It is difficult to see how the choices of which coincidences to add (or delete) from the presentation, and when, could be automated, especially in more general cases.

For our final investigations of the Todd-Coxeter process, we returned to the 'correct' presentation of E_3; that is, the three-generator, three-relator presentation obtained by substituting for r, s and t in E_2. To assess just how much more difficult than E_2 this presentation is, we attempted it over the subgroup $\langle r, s\rangle$. Although a few coincidences were obtained (about 3000 coincidences in 400 million cosets), no significant partial collapses were noted. It also has a significantly smaller throughput, at 41.9 seconds per million total cosets. However, when the two-generator subgroups were tried using the presentations obtained by substituting for r, s and t in E_2^a and E_2^b, then considerable progress was noted, although none of the enumerations completed successfully. For example, in the presentation derived from E_2^a, over $\langle r, s\rangle$, only about 382 million cosets remained active after defining a total of about 1275 million cosets.

8. Comparison with Knuth-Bendix

This family of presentations is better handled by Knuth-Bendix rewriting methods. The two packages RKBP (Rutgers Knuth-Bendix package, [**15**]) and KBMAG (Knuth-Bendix in monoids, and automatic groups, [**7**]) were used for some comparisons between the Todd-Coxeter and the Knuth-Bendix processes. For the memory requirements of the Knuth-Bendix programmes we quote the maximum total length of the rules (both sides) during a calculation. How this figure translates into a memory requirement depends on the data structure used to store the rules, and on any auxiliary structures used to speed up rule access. There is considerable scope for trading off speed against memory requirements; in fact, one of the design aims of RKBP (which is essentially an implementation of the work described in [**14**]) is to allow the manipulation of very large rewriting systems by using a very compact rule representation. There is less scope for such trade offs in the Todd-Coxeter process, so comparing memory requirements this way is perhaps a little unfair to coset enumeration; especially since large partial tables (which are the type of tables encountered in our tests) are typically quite sparse. However, it proved difficult to measure the amount of memory devoted to the rules in the Knuth-Bendix programmes and, as noted, this is to some extent under the user's control anyway.

One of the examples supplied with KBMAG is for E_2, and this example completes in 0.6 seconds on the SGI Origin 2000, with maximum rule length of 17070. E_2^*, the six-generator/six-relator version of E_2, completes in 0.1 seconds, with maximum rule length of 14659. The nine-generator/nine-relator form of E_3 completes in 0.5 seconds, with maximum rule length of 25422. The six-generator/six-relator form of E_3 completes in 1.7 seconds, with maximum rule length of 275156. As for the 'correct' three-relator/three-generator presentation of E_3, KBMAG completes in 1347.4 seconds, with maximum rule length of 14836607.

9. Conclusions

We have shown that proving E_2 trivial by coset enumeration is possible and that, although significant preliminary work is needed, the enumeration itself has very modest resource requirements. Interestingly, our best results are obtained by proving relations in the group, adding these to the defining relators, and restarting the enumeration; a technique borrowed from the Knuth-Bendix procedure. However, even with this technique, the Knuth-Bendix procedure outperforms Todd-Coxeter in terms of both space and time for the particular presentations of the trivial group considered here. Our work also suggests that for difficult enumerations in general, a combination of the Todd-Coxeter and

Knuth-Bendix processes, along with considerable experimentation (that is, intelligent guidance), may allow an enumeration to succeed.

References

[1] J.J. Cannon, L.A. Dimino, G. Havas and J.M. Watson, 'Implementation and analysis of the Todd-Coxeter algorithm', *Math. Comp.* **27** (1973), 463–490.

[2] G. Cooperman and G. Havas, 'Practical parallel coset enumeration', in *Workshop on High Performance Computing and Gigabit Local Area Networks*, (G. Cooperman, G. Michler and H. Vinck, Editors), Lecture Notes in Control and Information Sciences **226** (Springer-Verlag, Berlin, Heidelberg, New York, 1997), pp. 15–27.

[3] G. Havas, 'Coset enumeration strategies', in *ISSAC'91 (Proceedings of the 1991 International Symposium on Symbolic and Algebraic Computation)*, (Stephen M. Watt, Editor) (ACM Press, New York, 1991), pp. 191–199.

[4] G. Havas and C. Ramsay, 'ACE, Version 3 (1999)', Available as `http://www.csee.uq.edu.au/~havas/ace3.tar.gz`.

[5] G. Havas and C. Ramsay, 'Experiments in coset enumeration', (Technical Report 13, Centre for Discrete Mathematics and Computing, The University of Queensland, 1999).

[6] G. Higman, 'A finitely generated infinite simple group', *J. London Math. Soc.* **26** (1951), 61–64.

[7] D.F. Holt, 'KBMAG, Version 2.2 (1997)', Available via anonymous ftp from `ftp.maths.warwick.ac.uk`, in directory `/people/dfh/kbmag2`.

[8] D.E. Knuth and P.B. Bendix, 'Simple word problems in universal algebras', in *Computational Problems in Abstract Algebra* (Pergamon Press, Oxford, 1970), pp. 263–297.

[9] J. Leech, 'Coset enumeration on digital computers', *Proc. Cambridge Philos. Soc.* **59** (1963), 257–267.

[10] J. Leech, 'Coset enumeration', in *Computational Group Theory*, (Michael D. Atkinson, Editor) (Academic Press, London, New York, 1984), pp. 3–18.

[11] J. Neubüser, 'An elementary introduction to coset-table methods in computational group theory', in *Groups – St. Andrews 1981*, London Mathematical Society Lecture Note Series **71** (Cambridge University Press, Cambridge, 1982), pp. 1–45.

[12] B.H. Neumann, 'An essay on free products of groups with amalgamations', *Philos. Trans. Royal Soc. London Ser. A* **264** (1954), 503–554.

[13] B.H. Neumann, 'Proofs', *Math. Intelligencer* **2** (1979), 18–19.

[14] C.C. Sims, *Computation with finitely presented groups* (Cambridge University Press, Cambridge, 1994).

[15] C.C. Sims, 'RKBP, Version 1.3 (1995)', Available via anonymous ftp from `dimacs.rutgers.edu`, in directory `/pub/rkbp`.

[16] J.A. Todd and H.S.M. Coxeter, 'A practical method for enumerating cosets of finite abstract groups', *Proc. Edinburgh Math. Soc.* **5** (1936), 26–34.

Centre for Discrete Mathematics and Computing
Department of Computer Science and Electrical Engineering
The University of Queensland
Queensland 4072
Australia
e-mail: havas@csee.uq.edu.au
cram@csee.uq.edu.au