A MACHINE PROGRAM FOR COSET ENUMERATION

H. F. Trotter

(received January 15, 1964)

We are concerned with an algorithm for determining the index (when it is finite) of a subgroup H of a group K when K is specified by a finite set of generators and relations and H is specified as generated by a finite set of words in the generators of K. A systematic computational attack on the problem was discovered by Coxeter and Todd [1], and has proved to be a very useful tool in problems involving generators and relations in groups [2]. Although the method was not completely formalized it was clearly possible to convert it into a computer program, and this has been done by a number of people. Leech has given a survey of this work in [3], where further references may be found.

The author has written yet another such program, using a method which is logically much the same as Leech's. This paper consists of a formal description of the algorithm, a proof that it works, and some comments on the actual program. Leech presents a convincing argument for the effectiveness of the procedure in [3], but does not give a formal proof. Mendelsohn [4] has given a proof for a version of the algorithm which is closer to the original Todd-Coxeter method but differs somewhat from the version discussed here.

We interpret each step of the algorithm in group-theoretic terms, and are thus able to describe in such terms the point at which the computation terminates (see the proof of theorem 2). Theorem 1 in section 2 is an interesting by-product, which is presumably known, but does not appear to have been explicitly noted before.

Canad. Math. Bull. vol. 7, no. 3, July 1964.

https://doi.org/10.4153/CMB-1964-033-x Published online by Cambridge University Press

Section 1 contains a preliminary reformulation of the problem and section 2 develops the theoretical description of the algorithm. The necessary practical modifications of the process are discussed in section 3.

I am indebted to N. Mendelsohn for stimulating my interest in this problem, and for a prepublication copy of [4]. Correspondence with J. Leech regarding some examples used to test the program was most illuminating. The computer facilities used in this work were supported in part by the United States National Science Foundation, grant NSF-GP579.

Restatement of the problem in terms of free groups

Our problem may be stated informally thus: Find the index of H in K, where K is generated by $y_1, \ldots, y_n$ subject to the relations $r_1 = r_2 = \ldots = r_s = 1$, and H is generated by $h_1, \ldots, h_t$. Here the r_i and h_i are expressed in terms of the y_i. The following is a more formal restatement of the above:

Let F be a free group on the generators $x_1, \ldots, x_n$. Let $r_1, \ldots, r_s$ be words in F, let R be the set of all their conjugates in F, and let $[R]$ be the subgroup generated by R. Define K as $F/[R]$ and y_i as the image of x_i under the canonical map. Let $g_1, \ldots, g_t$ be elements of F, $h_1, \ldots, h_t$ their images in K, and H the subgroup generated by the h_1.

It is required to find the index of H in K.

Let G be the subgroup of F generated by $g_1, \ldots, g_t$ and R. Under the canonical map, the cosets of H in K correspond to the cosets of G in F, and the original problem is equivalent to finding the index of G in F. Theorem 2 in section 2 states precisely in what sense we solve the latter problem.

The algorithm in theory

The algorithm will be described in terms of operations on an array A of integers with $n+1$ columns and a varying number of rows. (In the theoretical discussion we admit the possibility of an infinite number of rows.) The columns are labelled from 0 to n, with columns 1 to n corresponding to the n generators $x_1, \ldots, x_n$ of F and column 0 corresponding to the identity element. For notational convenience we suppose the rows to be numbered consecutively from 1, but the order of the rows has no significance. The array need not be complete, i.e. some cells A_{ij} may be empty, but we assume that every row contains at least one number.

The following four conditions will be assumed to hold for <u>all</u> arrays.

(1) Any row with more than one entry has an entry in column 0.

(2) The integer 1 appears in the array.

(3) Any integer appearing in the array appears at least once in each column.

(4) No proper subset of the rows forms an array satisfying (3).

These are two further properties for which it is convenient to have names. We call an array <u>consistent</u> if

(5) No integer appears more than once in any column.

We call an array <u>complete</u> if

(6) No cell is empty.

An array which is both consistent and complete can be interpreted as a "multiplication table" defining a transitive representation of F by permutations of the integers appearing in the array. For an integer p, $p \cdot x_j$ is defined as the entry in column j of the row with p in column 0, while $p \cdot x_j^{-1}$

is the entry in column 0 of the row with p in column j.
Since F is free this specification of the action of the generators can be extended to give a representation of the whole group. Conditions (3), (5) and (6) ensure that the operations are well defined, and (4) is equivalent to transitivity of the representation. Given such an array A, we define S(A) as the subgroup of F which leaves the integer 1 invariant. The cosets of S(A) in F are in one-one correspondence with the different integers appearing in A, and so the index of S(A) in F is equal to the (possibly infinite) number of rows of A.

Any array satisfying (1) - (4) can be interpreted as partially specifying a representation on equivalence classes of the integers in A and defines a corresponding subgroup S(A). We proceed with the formal details.

A <u>link</u> in A is a pair (i, u) where i is a row number, $u = x_j$ or x_j^{-1}, and neither A_{i0} nor A_{ij} is empty. The <u>head</u> of (i, u) is A_{i0} if $u = x_j$ and A_{ij} if $u = x_j^{-1}$, while the <u>tail</u> of (i, u) is A_{ij} if $u = x_j$ and A_{i0} if $u = x_j^{-1}$. A sequence of links $c = (i_1, u_1) \ldots (i_m, u_m)$ is a <u>chain</u> in A if the head of each link after the first is equal to the tail of the preceding link. We say that c is a chain <u>from</u> p <u>to</u> q if p is the head of (i_1, u_1) and q is the tail of (i_m, u_m). The <u>inverse</u> of c is the sequence $(i_m, u_m^{-1}) \ldots (i_1, u_1^{-1})$ and is a chain if c is. The element of F represented by the word $u_1 \ldots u_m$ is said to be <u>covered</u> by c. Note that if c covers w then c^{-1} covers w^{-1}.

Define S(A) as the set of all w in F covered by some chain from 1 to 1 in A. Define p, q in A to be <u>equivalent</u> if there is a chain from p to q in A covering the identity element of F. It is completely straightforward to check that S(A) is a subgroup, and that distinct equivalence classes of integers correspond to distinct right cosets of S(A) in F. (Note that (2) and (3) imply that for any p appearing in A, there is a chain from 1 to p.)

We next describe the effect certain operations on A have on $S(A)$.

LEMMA 1. Suppose that the cell A_{ij} is empty and that p is an integer not occurring in A. Form A' by placing p in A_{ij} and for each column except the jth adding a new row with p in that column, so that condition (3) is satisfied. Then $S(A') = S(A)$.

Proof. Suppose $j \neq 0$. Then the only essential difference between A' and A is that A' contains the additional links (i, x_j) and (i, x_j^{-1}). Obviously, $S(A) \subset S(A')$. Conversely, take any w in $S(A')$ and let c be a chain from 1 to 1 in A' covering w and of minimal length. If either of the new links occurs in c the combination $(i, x_j)(i, x_j^{-1})$ must occur, since (i, x_j) is the only link in A' with p as tail and (i, x_j^{-1}) is the only one with p as head. An occurrence of $(i, x_j)(i, x_j^{-1})$ would contradict the minimality of c. Hence c is a chain in A and $w \in S(A)$. If $j = 0$, then by condition (1) there is a unique $k \neq 0$ with A_{ik} non-empty and a similar consideration of (i, x_k^{-1}) and (i, x_k) completes the proof.

LEMMA 2. Let $p < q$ be two integers in A and let c_p, c_q be chains in A from 1 to p and from 1 to q respectively. Form A' from A by replacing all occurrences of q by p. Then $S(A')$ is the subgroup generated by $S(A)$ and $w_p w_q^{-1}$ where w_p, w_q are covered by c_p, c_q respectively.

Proof. Let G be the subgroup generated by $S(A)$ and $w_p w_q^{-1}$. Any sequence of links which is a chain in A is also a chain in A' and so $S(A) \subset S(A')$. In A', $c_p c_q^{-1}$ is a chain from 1 to 1 covering $w_p w_q^{-1}$. Hence $w_p w_q^{-1} \in S(A')$ and $G \subset S(A')$. Conversely, suppose $w \in S(A')$ and $c = (i_1, u_1) \ldots (i_m, u_m)$ is a chain from 1 to 1 in A' which covers w. Considered as a sequence of links in A, c can fail to be a chain in A only because of breaks where the tail of one link is p and the head of the next is q, or vice versa. Let $b(c)$ denote the number of such breaks. (If $p = 1$, then c may not be from 1 to 1 in A. A trivial variation of the following argument will handle this possibility; for simplicity we disregard it.) Assume the inductive hypothesis that if w is any element of $S(A')$ covered by a chain c from 1 to 1 in A' such that $b(c) < s$, then $w \in G$. The hypothesis is true for $s = 1$, since if $b(c) = 0$ then c is a chain in A and $w \in S(A) \subset G$. Now consider a $w \in S(A')$ covered by a chain with $b(c) = s$. Suppose the first break occurs after the k-th link, and for definiteness suppose that the tail of (i_k, u_k) is p, and that the head of (i_{k+1}, u_k) is q. Consider the sequence $\bar{c} = c_1 c_2 c_3$ where $c_1 = (i_1, u_1) \ldots (i_k, u_k) c_p^{-1}$, $c_2 = c_p c_q^{-1}$ and $c_3 = c_q (i_{k+1}, u_{k+1}) \ldots (i_m, u_m)$. Now $\bar{c}$ covers w, so $w = w_1 w_2 w_3$, where w_i is covered by c_i. The sequence c_1 is a chain from 1 to 1 in A and so $w_1 \in S(A) \subset G$; $w_2 = w_p w_q^{-1} \in G$; c_3 is a chain from 1 to 1 in A' and $b(c_3) < s$ so by the inductive hypothesis, $w_3 \in G$. Hence $w \in G$ and the proof is complete.

COROLLARY. If p and q are equivalent in A, and A' is obtained by replacing q by p throughout A then $S(A') = S(A)$.

Proof. Let c_{pq} be a chain from p to q which covers the identity. Apply lemma 2 with c_p any chain from 1 to p and take $c_q = c_p c_{pq}$. Since $w_p = w_q$, $w_p w_q^{-1}$ is the identity and the result follows.

LEMMA 3. Given a finite array A, there is a finite algorithm for obtaining a consistent array B with $S(A) = S(B)$.

Proof. Suppose A is not consistent so that, say $A_{ij} = A_{kj}$. If either of the rows has only the single entry, it may be deleted without affecting $S(A)$; otherwise both A_{i0} and

A_{k0} are non-empty. For each column number $m \neq j$ proceed as follows:

(i) if $A_{im} = A_{km}$, or if both are empty, make no change,

(ii) if just one of A_{im}, A_{km} is empty copy the other entry into it,

(iii) if $A_{im} \neq A_{km}$, replace the larger number by the smaller throughout the array.

Operation (ii) clearly does not affect $S(A)$, and if (iii) applies then A_{im} is equivalent to A_{km} and by the corollary to lemma 1, $S(A)$ is unchanged. When these operations have been done for all columns the two rows i and k are identical, and one may be dropped. Thus from any inconsistent array A we obtain an equivalent array A' with one less row. If A is finite the process must lead to a consistent array in a finite number of steps.

Remark. In a consistent array no two distinct integers are equivalent.

LEMMA 4. Given an array A and a word $w \in F$, an array B may be constructed in a finite number of steps such that $S(A) = S(B)$ and for some q there is a chain from 1 to q in B which covers w.

Proof. The construction can be made by applying the operation of Lemma 1 not more than m times, where m is the length of w.

THEOREM 1. Given a finite set of words $u_1, \ldots, u_m$ in a finitely generated free group, there is a finite algorithm for determining the index of the subgroup they generate.

Proof. Let G_k, $k = 1, \ldots, m$ be the subgroup generated by the first k of the u_i and let G_0 consist of the identity. Start with an array A^0 which has $n + 1$ rows, a 1 in each row and column, and no other entries. Then $S(A^0) = G_0$. Suppose A^{k-1} has been constructed, with $S(A^{k-1}) = G_{k-1}$. Apply lemma 4 if necessary to obtain an array with a chain over u_k from 1 to p_k, and then replace p_k by 1 to obtain A^k. By lemma 2, $S(A^k) = G_k$. Finally, apply lemma 3 to obtain a consistent array B with $S(B) = G_m$.

If B is complete, G_m has finite index equal to the number of rows of B. On the other hand if B is not complete, the construction of lemma 1 leads to an array B' with $S(B') = G_m$. Examination of the construction shows that B' is also consistent and incomplete (unless $n = 1$, B' actually has more empty cells than B) so the same operation can be applied to B', and so on. Carrying out the process countably many times gives a complete consistent array, showing that G_m has infinite index.

We are now in a position to attack the original problem. Let G be the subgroup of F generated by words $g_1, \ldots, g_s$ and all conjugates of the words $r_1, \ldots, r_t$. Since the latter set is infinite we cannot simply apply theorem 1. We describe a process for constructing a sequence of consistent arrays $A^0, A^1, \ldots$ which may or may not terminate. We write G_k for $S(A^k)$.

Use the method of theorem 1 to construct A^0 so that G_0 is generated by $g_1, \ldots, g_s$, and then continue as follows:

(a) Take p as the smallest unprocessed integer in A^k. (Initially all integers are unprocessed and $p = 1$ will be the choice when $k = 0$.)

(b) Let c be a chain from 1 to p, and let w be the word it covers. Obtain a new consistent array B^{k+1} by the method of theorem 1 so that $S(A^{k+1})$ is generated by G_k and the elements $wr_1w^{-1}, \ldots, wr_tw^{-1}$. Whenever a new integer is introduced in the construction use an integer larger than any that has yet appeared; i.e. an integer eliminated by use of lemma 2 should not be reintroduced.

(c) If necessary, apply lemma 1 to fill all cells of the row with p in column 0, and column 0 of all rows with p in other columns. Take the resulting array as A^{k+1}. Then $G_{k+1} = S(A^{k+1}) = S(B^{k+1})$ is generated by G_k and $wr_1w^{-1}, \ldots,$ wr_tw^{-1}.

(d) Mark the integer p as processed, and increase k by 1.

(e) If the array contains any unprocessed integers, return to step (a). If it does not, stop.

The contention is that this procedure terminates if and only if G has finite index in F, and that if it does terminate then $G = S(A^f)$ where A^f is the final array. If the process terminates, then, in virtue of step (d), A^f must be complete and hence G_f has finite index. Since $G_f \subset G$, this shows that the process cannot terminate unless G has finite index. Conversely suppose G does have finite index. Then by the Nielsen-Schreier theorem [5] it is finitely generated. Each member of a finite set of generators is expressible in terms of the g's and a finite number of the conjugates of the r's; hence G is actually generated by the g's and some finite number of conjugates of the r's. Now observe that if w is any word covered by a chain from 1 to a marked integer in A^k, then all the conjugates wr_iw^{-1} are in G_k. Because of (d), for any $w \in F$ there will be for some k a chain in A^k over w from 1 to some integer q. Eventually either q itself will be marked or it will have been replaced by some smaller integer which is marked. Thus any finite set of conjugates of the r's will be in G_k for some sufficiently large k and consequently for some k, $G_k = G$ and is of finite index. This implies that A^k will be complete. None of the steps (a) - (e) can introduce new integers into a complete array, so in a finite number of steps the algorithm will terminate. At this point $A^f = A^k$ and $S(A^f) = G_k = G$. Thus we have proved

THEOREM 2. Given words $g_1, \ldots, g_s$ and $r_1, \ldots, r_t$ in a finitely generated free group, let G be the group generated by the g' s and the conjugates of the r' s. There is an algorithm which terminates if and only if G is of finite index in F. If the algorithm terminates, it determines the index of G in F.

The practical algorithm

A program called COSET has been written for the IBM 7090/94 computer which carries out a process logically equivalent to the one outlined in the preceeding section. A list of g' s and r' s is punched on cards and read into the computer. The output from a problem is either a statement of the index of the subgroup and of the permutation associated with each generator of F, or else a statement to the effect that computation led to an array too large for the storage capacity of the computer. In the program' s present form, n, the number of generators of F, may be at most 9, and the maximum number of rows which can be accommodated is about $29,000/(n + 2)$. An improved version is planned, which will have a capacity of about $29,000/(n + 1)$ rows.

Considerable effort went into making the program efficient in its use of time, and it does in fact work rapidly. For example, it found the 448 cosets of $\{A^2, A^{-1}B\}$ in the group with relations $A^8 = B^7 = (AB)^2 = (A^{-1}B)^3 = 1$ in under 6 seconds. (This example is proposed as a test problem in [3].)

The logical essentials of the algorithm were given in section 2, but for efficiency of operation a number of complications must be added. If w is covered by a chain from 1 to p, then finding (or creating) a chain from 1 to 1 over wrw^{-1} amounts to finding (or creating) a chain from p to p over r and this is the procedure actually followed. In Leech' s terminology, one "applies the coset p to the relation r".

The single array as described contains all the necessary information, but not all the information is readily accessible. For instance, to find $p \cdot x_j^{-1}$ requires a search of column j. Such searching is comparatively time-consuming for a computer, and it is necessary to avoid it by "double-entry bookkeeping" using additional columns from $n + 1$ to $2n$ with $A_{i,j+n}$ containing $A_{i0} \cdot x_j^{-1}$.

There is one feature of COSET which is not used in other programs, and which I believe contributes substantially to its speed. Column 0 is not explicit in any of these programs; in effect the row number (corresponding directly to the address of the row in storage) is understood to be the column 0 entry. Thus, when an application of lemma 2 calls for replacing all occurrences of one number by another, the replacement must actually be carried out. Further shifting of rows and consequent renumbering is needed to eliminate the gap left by the row corresponding to the deleted number. In COSET an extra column is maintained. Initially the entry in this column is set equal to the number of the row in which it appears. Whenever the program refers to a number it actually looks up the extra entry in the corresponding row. Consequently the effect of replacing q by p throughout the array is obtained simply by replacing the extra entry in row q by p. That row q has been effectively deleted is indicated by the fact that the extra entry in row q is no longer equal to q.

There is a consolidation routine which does actually rearrange rows and change numbers throughout the table, but it comes into play only at the end of the program (so that the printed permutations will deal with cosets numbered consecutively from 1 on), or at such times as the table threatens to exceed the available space (so that the space taken up by "deleted" rows may be made available for reuse).

REFERENCES

1. J.A. Todd and H.S.M. Coxeter, A practical method for enumerating cosets of a finite abstract group, Proc. Edinburgh Math. Soc. (2), 5 (1936), 26-34.

2. H.S.M. Coxeter and W.O.J. Moser, Generators and relations for discrete groups, Ergebnisse der Math. 14 (Springer; Berlin, 1957).

3. J. Leech, Coset enumeration on digital computers, Proc. Camb. Phil. Soc. 59 (1963), 257-267.

4. N.S. Mendelsohn, An algorithmic solution for a word problem in group theory, Dept. of Math. Publications, Univ. of Manitoba, 1963.

5. O. Schreier, Die Untergruppen der freien Gruppen, Hamb. Abh., 5 (1926), 161-183.

Princeton University