A Tietze Transformation Program

GEORGE HAVAS, P. E. KENNE, J. S. RICHARDSON
AND E. F. ROBERTSON

1 Introduction

A Reidemeister-Schreier program which yields a presentation of a subgroup H of finite index in a finitely presented group G is described in [3]. The program has two stages; first, Schreier generators and Reidemeister relators for H are computed, then the resulting presentation is simplified by eliminating redundant generators and by using a substring searching technique. The Tietze transformation program which we describe in this paper was originally designed to improve the simplification stage of that Reidemeister-Schreier program and now also forms part of the implementation of the modified Todd-Coxeter method [1]. The program described here is written in a reasonably portable superset of FORTRAN 66, and was available at the Symposium.

2 Program Description

The three main principles used by the Tietze program in simplifying the presentation are as follows.

(i) All relators of length 1 and non-involutory relators of length 2 are used to eliminate generators.

(ii) Long substrings of relators are replaced by shorter equivalent strings. First substring searching is performed. A relator r_1 is chosen (either by the program or by the user), and other relators are searched for a matching substring v in a cyclic permutation uv of r_1 and in a cyclic permutation wv of r_2 or its inverse, with the length of v greater than the length of u. Then, when a suitable match is found, the relator r_2 is replaced by a canonical representative of the shorter relator wu^{-1}. One substring replacement pass involves the application of this process with r_1 running through all relators.

COMPUTATIONAL GROUP THEORY
ISBN 0-12-066270-1

(iii) Redundant generators (i.e. generators which occur only once in some relator) are eliminated using relators with length greater than 2.

Notice that (i) and (ii) cannot increase the total length of the presentation but (iii) can, and probably does, increase the length.

The relators are stored as circular doubly-linked lists. This enables rapid access to all cyclic permutations of a relator and its inverse. A list of relators is maintained, sorted into ascending order (the canonical order defined in [3]). This assists the elimination of duplicate relators and also helps to reduce the work in substring searching. Information about a relator, such as its length and exponent, are maintained in a separate array.

A list of generators is also kept, containing for each generator such information as the total number of occurrences of the generator. This information is useful since it is often a good strategy to eliminate a generator which occurs infrequently.

3 Substring Searching

A substring search with two relators takes time dependent on the length of the longer relator but relatively independent of the length of the shorter relator, by virtue of the following observation. It suffices to check initially for occurrences of at most two letters of the shorter relator in the longer relator. If the short relator has unit exponent, any "long" common substring must contain the first letter of the short relator or a letter half way round that relator. If the short relator has non-trivial exponent then a long common substring must contain the first letter of the short relator. This makes the substring search process in the Tietze program take time dependent on N^2l, where N is the number of relators and l is the total relator length.

Substring replacement passes can be applied with potential beneficial effect after any significant alteration to a presentation. In particular, they are often effective after generator eliminations and after substring replacement passes which have led to changes. In calculations with lengthy presentations containing many hundreds of relators, the amount of substring searching may have to be curtailed to save time.

Two ways of reducing substring searching are to do several eliminations between substring replacement passes and to reduce the number of consecutive passes. The largest saving in length is usually made on the first pass and interrupting the search after one or two passes may save much time. Rather surprisingly, better final presentations are often obtained by making several eliminations between substring replacement passes, particularly when the redundant generators are being replaced by reasonably short words.

Sims pointed out, during his lectures at the Symposium, that substring

searching can be done in time proportional to Nl. Some interesting practical comparisons of string matching algorithms are made in [6].

4 Other Simplification Techniques

The latest version of the Tietze program contains several features to help simplify presentations in addition to (i), (ii) and (iii) described in §2. The first of these is similar to the substring replacement technique except that substitutions are only made for matching substrings of equal length. After equal length substitutions have been made, a substring replacement pass may again shorten the presentation. A separate technique deals with involutory generators.

Another feature allows the introduction of a new generator equal to the substring of length 2 (not a square) which occurs most often in the presentation. The generator from the relator of length 3 introduced, which occurs least often in the presentation, is then eliminated. This technique, like the others described in this section, seems to be most effective if used only in the final stages of reducing a presentation.

5 Applications

The Tietze transformation program is run with the user selecting an appropriate level of interaction. On the one hand the user may control the program closely (selecting which simplification techniques to apply in which order, monitoring the presentation as the program runs) or, alternatively, the user may leave it all to automatic running.

The program is mainly used to simplify Reidemeister-Schreier presentations, so that a resulting presentation for the subgroup may be input to a Todd-Coxeter coset enumeration program or a Reidemeister-Schreier program. Output may also be input to a nilpotent quotient algorithm or an abelian decomposition program. Typical applications of the version of the program discussed in this paper are described in [5] and [2].

Applications of an earlier version of the Tietze program are described in several papers, for example [4].

6 Test Examples

In order to give an indication of the improvements that the Tietze program gives, we examine the same 6 test examples discussed in [3].

(a) $G_1 = \langle a, b | a^3 = b^6 = (ab)^4 = (ab^2)^4 = (ab^3)^3 = 1, ab^2a^2b^2ab^2 = b^2ab^2a^2b^2a \rangle$,

$H_1 = \langle a, b^2 \rangle$, $|G_1 : H_1| = 26$.

(b) $G_2 = \langle a,b | a^4 = b^4 = (ab)^4 = (a^{-1}b)^4 = (a^2b)^4 = (ab^2)^4 = (a^2b^2)^4 = [a,b]^4 = (a^{-1}bab)^4 = 1 \rangle$,
$H_2 = \langle a, b^2 \rangle$, $|G_2 : H_2| = 64$.

(c) $G_3 = \langle a,b,c | a^{11} = b^5 = c^4 = (bc^2)^2 = (abc)^3 = (a^4c^2)^3 = b^2c^{-1}b^{-1}c = a^4b^{-1}a^{-1}b = 1 \rangle$,
$H_3 = \langle a,b,c^2 \rangle$, $|G_3 : H_3| = 12$.

(d) $G_4 = \langle a,b,c | a^{11} = b^5 = c^4 = (ac)^3 = b^2c^{-1}b^{-1}c = a^4b^{-1}a^{-1}b = 1 \rangle$,
$H_4 = \langle a,b,c^2 \rangle$, $|G_4 : H_4| = 12$.

(e) $G_5 = \langle a,b,c | a^3 = b^7 = c^{13} = (ab)^2 = (bc)^2 = (ca)^2 = (abc)^2 = 1 \rangle$,
$H_5 = \langle ab, c \rangle$, $|G_5 : H_5| = 42$.

(f) $G_6 = \langle a,b,c | a^3 = b^7 = c^{14} = (ab)^2 = (bc)^2 = (ca)^2 = (abc)^2 = 1 \rangle$,
$H_6 = \langle ab, c \rangle$, $|G_6 : H_6| = 78$.

The notation in Tables 1 and 2 is:

a = number of generators, b = number of relators
c = length of longest relator, d = total length of relators
(where length ignores overall exponent on relators);
I = Reidemeister-Schreier presentation (for which we omit column d),
II = final output presentation from Havas's program, [3].
III = output from Tietze using alternate eliminations and long substring replacement automatically,
IV = output from Tietze using all simplification techniques interactively.

Table 1

	I			II				III				IV			
	a	b	c	a	b	c	d	a	b	c	d	a	b	c	d
H_1	27	156	14	2	29	168	1664	2	9	12	44	2	7	8	26
H_2	65	576	16	2	67	204	2690	2	16	14	82	2	5	4	10
H_3	25	96	15	4	22	111	559	3	16	26	125	2	8	12	40
H_4	25	72	11	3	10	21	107	3	10	10	66	3	8	6	21
H_5	85	294	13	3	13	65	237	3	10	14	76	2	3	2	4
H_6	157	546	14	2	12	66	374	2	7	9	22	2	3	2	4

By way of examples, we quote the final presentations of H_1 and H_2 obtained from Tietze:

$H_1 = \langle a,b | a^4 = (ab^{-1})^3 = abab^{-2}a^{-1}b^{-2} = (ab^{-2})^3 = (a^2b^{-3})^3 = (aba^{-1}b^{-1})^4 = (ab^2)^6 = 1 \rangle$,
$H_2 = \langle a,b | a^4 = (ab)^2 = b^4 = (ab^{-1})^4 = (a^2b^2)^4 = 1 \rangle$.

When run automatically with all simplification techniques, the Tietze program will produce, in general, presentations of length somewhere between those quoted under III and IV although, for H_5, the automatic run produces the "best possible" result,

$$H_5 = \langle a, b \,|\, a^2 = b^2 = (ab)^{13} = 1 \rangle.$$

For certain groups, considerably better presentations can be obtained in an interactive run by selecting the order in which the redundant generators are eliminated. Examples studied in [2] typify this, where a presentation for $H(n) = \langle a, b^2 \rangle$ of index $2n+3$ in

$$Y(n) = \langle a, b \,|\, (abab^{-1})^n = ba^{-1}bab^{-1}a, ab^2a^{-1}ba^2b^{-1} = 1 \rangle$$

is found for $n = 5, \pm 7, \pm 8, \pm 10$; this presentation being used again as input to Reidemeister-Schreier and Tietze programs to find $|Y(n)|$ in each case. In Table 2, we give two examples of the performance of Tietze for $n = -8$ and $n = 8$.

Table 2

	III				IV			
	a	*b*	*c*	*d*	*a*	*b*	*c*	*d*
$H(-8)$	2	14	131	807	2	2	27	34
$H(8)$	4	22	110	881	2	4	38	99

References

1. Arrell, D. G. and Robertson, E. F. (1983). A Modified Todd-Coxeter Algorithm, (these Proceedings).
2. Campbell, C. M. and Robertson, E. F. Some problems in group presentations, *J. Korean Math. Soc.* (to appear).
3. Havas, G. (1974). "A Reidemeister-Schreier program", Proc. Second Internat. Conf. Theory of Groups (Canberra, 1973), Lecture Notes in Mathematics, Vol. **372**, 347–356. Springer-Verlag, Berlin.
4. Havas, G. and Richardson, J. S. (1983). Groups of exponent five and class four, *Comm. in Alg.* **11**, 287–304.
5. Havas, G. and Robertson, E. F. (1983). Two Groups which Act on Cubic Graphs, (these Proceedings).
6. Smit, G. De V. (1982). A comparison of three string matching algorithms, *Software Practice and Experience* **12**, 57–66.