Mathematics 7350

The Todd–Coxeter procedure

Ken Brown, Cornell University, September 2013

The Todd–Coxeter procedure is a systematic way of trying to find the order (and often the structure) of a group given by generators and relations, provided the group is finite. More generally, we can try to find the index $|G : H|$ of a subgroup H of finite index, even if G itself is infinite. The problem of finding the order is the special case $H = \{1\}$. A good introductory reference is Neubüser [9] (supplemented by [3].) The original paper of Coxeter and Todd [4] is also quite readable. For a more detailed treatement and further references, see Holt [8, Chapter 5], in which all algorithms are written out completely in pseudocode. Our treatment will be somewhat less formal.

We will describe the method in terms of Schreier graphs, so we begin by recalling what those are.

1. Schreier graphs

If σ is a permutation of a set X, you can picture σ by drawing a directed graph with X as vertex set and with an edge from x to $\sigma(x)$ for each $x \in X$. Thus each vertex has exactly one edge coming in and one going out. Note that the graph breaks up into connected components, one for each cycle in the cycle decomposition of σ.

If you want to picture two or more permutations of X simultaneously, you can use a different color (or label) for each one of them. Then each vertex has exactly one edge of each color coming in and one going out.

If you want to picture a whole group of permutations, you don't have to draw them all; it's enough to draw the pictures of a set of generators of the group. You can then read off the action of any element of the group by writing it as a word in the generators and their inverses and following the corresponding path in the graph. Note that this works much better for right actions than for left actions, since you can follow the path step by step as you read the word from left to right. We will therefore use right actions in what follows. In addition, we will use exponential notation for the action. Thus x^g denotes x acted on by g.

2. The Todd–Coxeter procedure: Example

Suppose G is a group given by generators and relations and H is a subgroup of finite index. The idea of the Todd–Coxeter procedure is to try to figure out what the Schreier graph of the (right) action of G on $H\backslash G$ looks like. What we do is make up names for cosets (typically 1,2,...) as we go along, and draw in arrows for the action of the generators as we discover them. The procedure stops when the graph is *complete*. This means:

- Every vertex has exactly one edge of each color coming in and one going out.
- The graph shows that all the relations are satisfied.

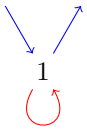
I will illustrate the method with the following example

$$G := \langle\, a, b \,;\, a^3 = b^3 = (ab)^2 = 1 \,\rangle\,.$$

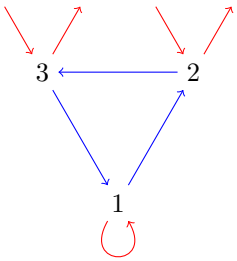
Let $H := \langle a \rangle \leq G$, and consider the action of G on $X := H\backslash G$. We will use red arrows to denote the action of a and blue arrows to denote the action of b. When the Schreier graph is complete, every red-red-red path will have to be closed as a result of the relation $a^3 = 1$. Similarly, every blue-blue-blue path will have to be closed and every red-blue-red-blue path will have to be closed. So if we see at some stage a subgraph consisting of two consecutive arrows of the same color, then we can fill in a third arrow to close the triangle. Similarly, if we see a subgraph consisting of three sides of a red-blue-red-blue square, then we can fill in the fourth side. Note that it doesn't matter what color the fourth side is, since a cyclic permutation of a relator is again a relator.

We now try to construct the Schreier graph.

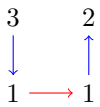
Step 1. Record the fact that there is an element of X (called "1") fixed by H. The two short blue arrows are reminders that we will eventually have to have a blue arrow coming in and a blue arrow coming out. But we don't yet have names for the vertices at the other ends.



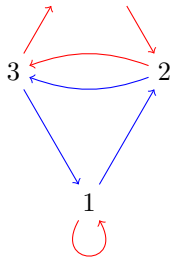
Step 2. We now check our three relations at the vertex 1. (i) The relation $a^3 = 1$ holds trivially (i.e., the red-red-red path starting at 1 is closed). (ii) In order to check the relation $b^3 = 1$, we introduce the names $2 := 1^b$ and $3 := 2^b$. The relation then tells us that we can close the blue triangle, yielding



Again we have drawn short arrows at the two new vertices as reminders of what is missing. (iii) Finally, we check the relation $abab = 1$ at vertex 1, i.e., we look for a red-blue-red-blue square starting at 1. Three sides of that square are already present:

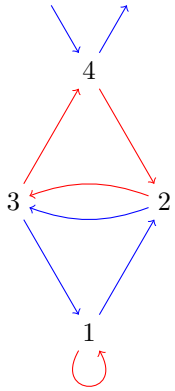


As noted above, we can fill in the square, i.e., we can deduce that $2^a = 3$. So our graph now looks as follows:



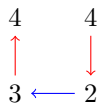
It is complete at vertex 1.

Step 3. We now move on to vertex 2 and check the three relations. (i) To check $a^3 = 1$, we define $4 := 3^a$ and then close the red triangle. This yields:

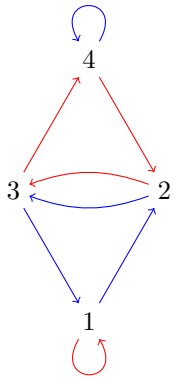


(ii) The relation $b^3 = 1$ already holds at vertex 2. (iii) The relation $abab = 1$ also holds. So the graph is complete at vertex 2.

Step 4. At vertex 3, the relations (i) and (ii) already hold. But in checking relation (iii), we see that we have three sides of the red-blue-red-blue square:



We fill in the fourth side and deduce that $4^b = 4$. Our graph is now



It is complete at 3.

Step 5. Finally, we move to vertex 4 and check that all three relations hold. The graph is complete.

One can now conclude that $|G : H| = 4$ and that the graph above really describes the action of G on X. (The essential content of this is that cosets with different labels really are distinct.) To prove this, let $\widetilde{X}$ be the abstract 4-element set $\{1, 2, 3, 4\}$, where the elements are viewed as labels, not as cosets. The graph enables us to define an action of G on $\widetilde{X}$. Indeed, it exhibits permutations associated to the generators of G, and we've verified that these permutations satisfy the defining relations of G; so we get a well-defined action. Moreover, the action is transitive because the graph is connected, and clearly H stabilizes 1.

There is a map $p\colon \widetilde{X} \to X$ that takes each label to the corresponding coset, and this map is G-equivariant because of the way we constructed the graph. (Every edge that was drawn described the known action of a generator of G on an element of X.) It follows that the stabilizer of $1 \in \widetilde{X}$ can't be bigger than H and hence that p is an isomorphism of G-sets. This proves the claim.

We now know that H has order 3 (i.e., that the generator a of G is not trivial), since it acts nontrivially on X. Hence G has order 12. In fact, G is isomorphic to the alternating group A_4. To see this, note that the action of G on X gives us a homomorphism $G \to A_4$, and it is easy to check that the images of a and b generate A_4. Our assertion now follows from a counting argument.

Remark. This example was simpler than some, in that all the names we introduced $(1, 2, \dots)$ turned out to represent different cosets. A more common situation is that, at various stages of the procedure, two cosets with different names are discovered to be identical. This situation is known in the literature as a *coincidence*. When we discover a coincidence, the graph we have constructed up to that point will still map to the true Schreier graph, but the map will not be injective. We therefore replace the graph with a quotient graph before continuing. We will give an example of this in Section 4.

Coincidences are inevitable in general. [Think about presentations of the trivial group, for instance.]

3. Relator tables and scanning

The process of checking that a relator is satisfied at a vertex is called *scanning*. In the example above, this involved looking for red-red-red triangles, blue-blue-blue triangles, and red-blue-red-blue squares. In more complicated examples it is often helpful to use *relator tables*, one for each relator, to show the verification that the relations are satisfied. The completed relator table for *abab* in our example above is

a	b	a	b	
1	1	2	3	1
2	3	1	1	2
3	4	4	2	3
4	2	3	4	4

Each row starts and ends with the same coset, and the entries show the action of the various generators that occur in the relator. For example, the entries of the last row come from the equations $4^a = 2$, $2^b = 3$, and so on. The fact that the row

starts and ends with 4 shows that 4 is fixed by *abab*. We say that coset 4 "scans correctly" under *abab*.

To illustrate the use of relator tables, let's redo our example from that point of view. As in Step 1 above, we start with coset 1 satisfying $1^a = 1$. Scanning 1 under each of the relators gives

a	a	a	
1	1	1	1

b	b	b	
1			1

a	b	a	b
1	1		1

Only the *aaa* scan is complete at this point.

In order to make progress, let's define $2 := 1^b$ to get

a	a	a	
1	1	1	1
2			2

b	b	b	
1	2		1
2			2

a	b	a	b
1	1	2	1
2		1	2

Note that the entries we made in the second row of the *abab* table were obtained by scanning from right to left. In general, we try to scan from left to right and from right to left until the row is filled.

Now let's define $3 := 2^b$ in order to fill the gap in the first row of the *bbb* table. Since the next entry in that row is 1, we are able to make the *deduction* $3^b = 1$. (This is the analogue, from the point of view of relator tables, of closing the blue triangle in Step 2 of Section 2.) The relator tables become

a	a	a	
1	1	1	1
2			2
3			3

b	b	b	
1	2	3	1
2	3	1	2
3	1	2	3

a	b	a	b
1	1	2	1
2		1	2
3		2	3

We get the 3 in the first row of the *abab* table by scanning from right to left. This fills the row and gives us the deduction $2^a = 3$. This deduction allows us to go one step further in scanning the second row of the *aaa* table, leaving a single gap that can be filled by defining $4 := 3^a$. At this point it is easy to complete all scans, and the procedure terminates as in Section 2.

4. Coincidences: Example

We briefly mentioned coincidences at the end of Section 2. Here's an example. Consider the group

$$G := \langle\, a, b \,;\, aba^{-1} = b^2,\ bab^{-1} = a^2 \,\rangle$$

and the trivial subgroup H. Thus we are trying to work out the translation action of G on itself.

Start with a coset 1 and make successive definitions so that the first relator (written as $aba^{-1}b^{-1}b^{-1}$) scans completely: $2 := 1^a$, $3 := 2^b$, $4 := 3^{a^{-1}}$, and $5 := 4^{b^{-1}}$. We deduce that $5^{b^{-1}} = 1$, and the Schreier graph so far is a pentagon. [Draw it!] At this point coset 5 scans completely under the second relator, and we deduce $1^{a^{-1}} = 5$, allowing us to add a red edge from 5 to 1. Here are the relator

tables, where the uppercase letters denote the inverses of the generators:

a	b	A	B	B	
1	2	3	4	5	1
2				3	2
3					3
4	3				4
5	1	5		4	5

b	a	B	A	A	
1	5	1		2	1
2	3				2
3					3
4				3	4
5	4	3	2	1	5

Now define $6 := 1^{b^{-1}}$; this fills the gap in the first row of the second relator table and gives us the deduction $2^a = 6$.

At this point we start to get coincidences as we do further scans. For example, the second row of the first relator now scans completely, giving the deduction $5^{b^{-1}} = 3$. When we try to add the blue edge from 3 to 5 to the Schreier graph, we see that we already have a blue edge from 1 to 5. Hence we have obtained the coincidence $3 = 1$.

As we noted at the end of Section 2, we now want to replace our tentative Schreier graph Γ by a quotient graph that reflects the coincidence. This graph should still have the formal properties of a subgraph of a Schreier graph (at most one edge of each color entering or leaving a vertex). So we need to work out the equivalence relation on the vertices forced by this requirement and by the coincidence $3 = 1$.

Our convention in working with equivalence relations on vertices is that we always represent an equivalence class by its smallest element. We will also find it convenient to construct a function $p\colon \{1, \dots, 6\} \to \{1, \dots, 6\}$ such that $p(x)$ is equivalent to x for all x, and $p(x) \le x$ for all x with equality if and only if x is the representative of its equivalence class. We can think of p as an approximation to the quotient map; we obtain the true quotient map by iterating p until it stabilizes.

To get started, set $p(3) := 1$ and $p(x) := x$ if $x \neq 3$. At this stage our quotient has 5 vertices, represented by $1, 2, 4, 5, 6$. You might find it helpful to write "1" next to vertex 3 as a reminder that $p(3) = 1$.

Now consider the edges impinging on 3 in Γ. We have a blue edge $2 \longrightarrow 3$, which should become a blue edge $2 \longrightarrow 1$ in the quotient. But we already have a blue edge entering 1, which starts at 6. So we have obtained a new coincidence $6 = 2$, which we will need to process as soon as we finish with vertex 3. At this point we erase the edge $2 \longrightarrow 3$ in our picture and we redefine $p(6) := 2$ (and write "2" next to 6) to reflect that 6 is equivalent to 2. We also put 6 in a queue of cosets that have died but still need to be processed. (We will have to deal with the edges impinging on 6.)

Continuing with vertex 3, we have a red edge $4 \longrightarrow 3$. This yields the coincidence $5 = 4$, since there is already a red edge $5 \longrightarrow 1$. So we erase the edge $4 \longrightarrow 3$, redefine $p(5) := 4$, and add 5 to the queue. At this point we can erase vertex 3 if we want.

Now we process vertex 6, for which $p(6) = 2$. We change the blue edge $6 \longrightarrow 1$ to a blue edge $2 \longrightarrow 1$. The red edge entering 6 wants to change to a red edge entering 2. But we already have a red edge $1 \longrightarrow 2$, so there is yet another coincidence, $2 = 1$. We therefore erase the edge $2 \longrightarrow 6$, redefine $p(2) := 1$, add 2 to the queue, and (optionally) erase vertex 6.

Continuing in the way to process the vertices in the queue, we quickly find that the quotient graph consists of the single vertex 1, with a red loop and a blue loop. The Todd–Coxeter procedure has terminated, proving that G is the trivial group.

Coincidence processing is the most complicated part of the Todd–Coxeter procedure. I hope the method is more-or-less clear from this example, but I will re-explain it in Section 6 in different language.

Remark. There is a faster way to deduce that G is the trivial group via Todd–Coxeter. Instead of taking H to be the trivial subgroup, let $H := \langle a \rangle$. The Schreier graph collapses to the trivial graph after only a few steps. Thus $G = H$ and is therefore cyclic, hence abelian. But as soon as we know that G is abelian, the defining relations immediately imply that G is trivial.

5. Further examples

If you want to try some examples on your own, here are a few where the Todd–Coxeter method reveals the structure fairly easily:

- $\left\langle a, b \,;\, ab^2a^{-1} = b^3,\ ba^2b^{-1} = a^3 \right\rangle$ is the trivial group.
- $\left\langle a, b \,;\, a^3 = b^2 = (ab)^4 = 1 \right\rangle \cong S_4$, the symmetric group of degree 4.
- $\left\langle a, b \,;\, a^3 = b^5 = (ab)^2 = 1 \right\rangle \cong A_5$, the alternating group of degree 5.
- $\left\langle a, b \,;\, a^2 = b^2 = (ab)^2 \right\rangle \cong Q_8$, the quaternion group of order 8.

The last one is perhaps the most surprising. Note that there's no "$= 1$" at the end of the relations. So it's not even obvious that a and b have finite order. But somehow the relations force them to have order 4.

Finally, we briefly discuss the last example given in the original Todd–Coxeter paper [4]. There is a known group G_0 of order 576 generated by three elements a, b, c that satisfy the relations

$$a^3 = b^2 = c^2 = (ab)^4 = (ac)^2 = (bc)^3 = 1. \tag{1}$$

The goal is to prove that these are defining relations for G_0. To this end we denote by G the abstract group *defined* by the presentation with three generators subject to the relations (1). Thus we have a surjection $G \twoheadrightarrow G_0$, which will be an isomorphism if we can show that $|G| \leq 576$. Let $H := \langle a, b \rangle \leq G$. Then H is a quotient of S_4 in view of the second example above, so $|H| \leq 24$. A straightforward run of the Todd–Coxeter procedure shows that $|G : H| = 24$, so we conclude that

$$|G| = 24 \cdot |H| \leq 24^2 = 576,$$

as required.

6. Coset tables and coincidence processing

Most of the literature on the Todd–Coxeter procedure uses *coset tables* instead of Schreier graphs. A coset table lists the cosets and displays the action of the generators and their inverses. In other words, it lists the vertices of the Schreier graph and displays the endpoints of all outgoing and incoming arrows. For example,

the complete coset table for the example treated in Section 2 is

	a	A	b	B
1	1	1	2	3
2	3	4	3	1
3	4	2	1	2
4	2	3	4	4

Each row corresponds to a coset label, and each column corresponds to a generator or its inverse. We are still using the convention that uppercase letters denote the inverses of the generators. Row 2, for example, shows that $2^a = 3$, $2^{a^{-1}} = 4$, $2^b = 3$, and $2^{b^{-1}} = 1$. This is the same information that we would read off from the Schreier graph by looking at the arrows entering and leaving vertex 2.

We will denote by X the set of coset labels; thus X indexes the rows of the coset table. Both X and the coset table change during the Todd–Coxeter procedure. For brevity, I will often call the elements of X cosets. Recall that there might be coincidences, so the actual set of cosets $H\backslash G$ that we are trying to enumerate may be a quotient of the final X (assuming that the procedure terminates). Thus we will maintain an equivalence relation on X throughout the procedure.

As in Section 4, we will always represent an equivalence class by its least element, and we will use a function $p\colon X \to X$ such that $p(x)$ is equivalent to x and $p(x) \le x$, with equality if and only if x is the representative of its equivalence class. We call a coset $x \in X$ *live* if $p(x) = x$ and *dead* otherwise.

The dead cosets are those that I said could optionally be erased in the example in Section 4. The live cosets represent the equivalence classes and, if the procedure terminates, they correspond to the actual cosets. Whenever we make definitions or scan a coset, we deal only with live cosets. The dead cosets come into play only when we are processing a coincidence, since we will often have a queue of dead cosets that still need to be processed. At the end of the coincidence processing, it is harmless to remove the dead rows from the coset table.

We will denote by $\mathrm{Rep}(x)$ the representative of the equivalence class of x; this may change during coincidence processing. Recall that we can compute $\mathrm{Rep}(x)$ by iterating p until it stabilizes.

We now explain how coincidence processing works, in the language of coset tables; see Holt [8, Section 5.1.3] for more details, including a complete proof that the algorithm does what it is supposed to do. (The proof is routine but sightly tedious.)

Suppose we obtain a coincidence $x = y$, where x and y are live cosets and $x < y$. The first step is to merge the equivalence classes of x and y. In this case that simply means setting $p(y) := x$ and putting y into an initially empty queue of dead cosets that have to be processed. ("Processing" a dead coset will be explained below.) In general, if we speak of merging cosets x and y that are not necessarily live, we mean the following: Consider $x' := \mathrm{Rep}(x)$ and $y' := \mathrm{Rep}(y)$. We may have $x' = y'$, in which case there is nothing to do. Otherwise, redefine $p(y') := x'$ if $x' < y'$ or $p(x') := y'$ if $x' > y'$, and add the larger of x', y' to the queue.

Now let's spell out how to process an element v in the queue. We want to transfer all the information we have about v to information about $\mathrm{Rep}(v)$. When we're done, v should no longer appear in any live row of the coset table. Let S be the set of generators of G and their inverses. Consider the elements $s \in S$ one by

one in some order. We don't do anything unless v^s is defined in the coset table, say $u = v^s$. [You may find it helpful to draw pictures with labeled edges in order to keep track of the notation as we proceed.] Then we also have $u^{s^{-1}} = v$ in the table. The first thing we do is delete that entry, since we don't want v to remain in the u-row. The reader can check that an equivalent entry will be put back in later.

In principle, we would now like to insert entries $\bar{v}^s = \bar{u}$ and $\bar{u}^{s^{-1}} = \bar{v}$ into the table, where $\bar{v} := \mathrm{Rep}(v)$ and $\bar{u} := \mathrm{Rep}(u)$. But we may already have an entry for $\bar{v}^s$ in the table, say $\bar{v}^s = w$. In this case we simply merge $\bar{u}$ and w. Or we may already have an entry $w^{s^{-1}} = \bar{v}$ in the table, in which case we again merge $\bar{u}$ and w. The remaining possibility is that neither $\bar{v}^s$ nor $\bar{u}^{s^{-1}}$ has been defined; in this case we insert the desired entries $\bar{v}^s = \bar{u}$ and $\bar{u}^{s^{-1}} = \bar{v}$ into the table.

After we finish considering each $s \in S$, we remove v from the queue and move on to the next element, continuing until the queue is empty.

7. Summary of the Todd–Coxeter procedure

We are given a finitely presented group G and a subgroup H generated by finitely many elements of G, explicitly given as words in the generators of G. We denote by S the set of generators of G and their inverses. And we denote by X the set of coset labels at any given stage of the procedure. Let X_0 be the set of live cosets.

What we have presented so far, mostly by means of examples, is a nondeterministic procedure, with many choices as to how to proceed at each stage:

(1) We can make a definition $x^s = y$, where $x \in X_0$, $s \in S$, and the coset table does not currently have in entry for x^s. We always take y to be the first natural number that has not yet been used as a label, and we always add $y^{s^{-1}} = x$ along with $x^s = y$. [In terms of graphs, we have added a new vertex and a new labeled, directed edge.]
(2) We can scan a coset under a relator if it is possible to get further than the last time the same scan was done. If the scan completes, we get a deduction. We enter that deduction into the coset table unless it conflicts with an existing entry. In that case we have a coincidence (which we process).
(3) We can scan coset 1 under a generator of H if it is possible to get further than the last time the same scan was done. Again, we get a deduction if the scan completes, and we may get a coincidence (which we process).

In practice, we usually start the procedure by "scanning and filling" coset 1 under all generators of H. This means that we make a definition at the end of each scan to allow the scan to proceed further, until the scan completes successfully. The remainder of the procedure consists of steps of types (1) and (2).

A run of the Todd–Coxeter procedure (with some sequences of choices) is said to *terminate* if after finitely many steps, all live rows are completely filled in, all live cosets scan correctly under the relators, and coset 1 scans correctly under the generators of H. In other words, none of the three steps above can be taken. If this happens, then the argument used in the example of Section 2 shows that the live rows of the resulting coset table really describe the action of G on $H\backslash G$ (which is therefore finite).

Conversely, if $|G : H|$ is finite, we will show that the procedure terminates, provided we make our choices in a reasonable way.

Theorem. *Consider a run of the Todd–Coxeter procedure in which the choices are made so that the following conditions are guaranteed to hold:*

(a) For each coset label x that is introduced, either x will eventually die or else x^s will eventually be defined for all $s \in S$.
(b) Coset 1 will eventually scan correctly under every generator of H.
(c) Each coset label x that is introduced and does not die will eventually scan correctly under every relator.

If $|G : H|$ is finite, then the procedure terminates.

Proof. It follows from (a) that every row of the coset table eventually stabilizes and that the live rows of the resulting limiting table are completely filled in. Indeed, once an entry appears in a row, the only way it can change is if it decreases as a result of coincidence processing. And it cannot decrease more than finitely many times. Let X_∞ be the (possibly infinite) set of coset labels; these index the rows of the limiting table. In view of conditions (b) and (c), the argument that we used in Section 2 implies that the set $X_{\infty,0}$ of live cosets in X_∞ can be identified with $H\backslash G$ and is therefore finite. The rows corresponding to $X_{\infty,0}$ will therefore exist in their final form after finitely many steps. The theorem will follow if we can show that, at this stage, $X_0 = X_{\infty,0}$. In other words, every coset that will eventually die is already dead.

Suppose this is false, and let x be the first element of $X_0 \smallsetminus X_{\infty,0}$. I claim that x must occur in some row coresponding to an element of $X_{\infty,0}$, contradicting the fact that those rows have stabilized. This was certainly the case when x first appeared, since it was defined as y^s for some live $y < x$ and some $s \in S$. Now y may have later died, but then x would have been entered into some earlier live row as a result of coincidence processing. Repeating this argument, we see that x still occurs in a live row preceding the x-row. The label of this row is necessarily in $X_{\infty,0}$ by definition of x, proving the claim. □

8. Strategies

There are many possible strategies one can use in order to make sure that the hypotheses of the theorem are satisfied. These are discussed in detail in the references cited at the beginning of this handout. Here are three of them.

The strategy that we followed in Section 2 is often called the *HLT* method (after Haselgrove, Leech, and Trotter), or the *relator-based* method. We start by scanning coset 1 under the generators of H, making definitions, if necessary, to force the scan to complete. In the terminology introduced in the previous section, we scan and fill coset 1. We now proceed to consider the live cosets $x \geq 1$ in numerical order, scanning and filling under all the relators. Coset x might die as a result of coincidence processing after one of the scans, in which case we omit any further scans (and optionally erase row x), and we move on to the next live coset. Otherwise, we make further definitions to fill the x-row before continuing.

There is a variant of HLT, called *HLT plus lookahead*, which can be useful if we want to keep the coset table from getting too big and exhausting our computer's memory. Suppose we are working at coset x and the number of live cosets has exceeded some pre-determined threshold, possibly determined by the computer's available memory. Instead of scanning and filling, we pause and look ahead at later cosets in the hope of discovering coincidences. To this end, we scan (without filling)

all live cosets under all the relators. We then remove the dead rows, and we resume the HLT procedure at coset x if we have succeeded in recovering some space.

A third method in common use is the *Felsch* method or the *coset table–based* method. Instead of forcing scans to complete, we focus instead on filling the table. We still proceed from coset to coset, but we don't necessarily scan the current x under all the relators. [We do, however, start by scanning and filling 1 under all generators of H.] Instead, we make new definitions to fill in the x-row, and after each definition we scan *all* cosets (without filling) that could possibly be affected by the definition we've just made.

9. Software

Many computer algebra systems, including GAP [7] and Magma [1], incorporate the Todd–Coxeter procedure. GAP has, in addition to its built-in procedure, two special-purpose packages, called ACE [6] ("Advanced Coset Enumerator") and ITC [5] ("Interactive Todd–Coxeter"). The latter has a graphical interface and lets you control the process step by step, experimenting with different strategies. GAP and Magma are both installed on the Math Department computer system and can also be installed on your personal computers. GAP is free but not especially easy to install. Magma requires a license; see Steve Gaarder for details.

References

[1] Wieb Bosma, John Cannon, and Catherine Playoust. "The Magma algebra system. I. The user language". In: *J. Symbolic Comput.* 24 (1997). Computational algebra and number theory (London, 1993), pp. 235–265. URL: http://dx.doi.org/10.1006/jsco.1996.0125.

[2] C. M. Campbell and E. F. Robertson, eds. *Groups—St. Andrews 1981.* Revised. Vol. 71. London Mathematical Society Lecture Note Series. Selected papers from the International Conference held at the University of St. Andrews, St. Andrews, July 25–August 8, 1981. Cambridge: Cambridge University Press, 2007, pp. xiv+374. URL: http://dx.doi.org/10.1017/CBO9780511661884.

[3] Colin M. Campbell, George Havas, and Edmund F. Robertson. "Addendum to: "An elementary introduction to coset table methods in computational group theory"". In: [2], pp. 361–364. URL: http://dx.doi.org/10.1017/CBO9780511661884.028.

[4] H. S. M. Coxeter and J. A. Todd. "A practical method for enumerating cosets of a finite abstract group". In: *Proc. Edinb. Math. Soc. (2)* 5 (1936), pp. 26–34. URL: http://dx.doi.org/10.1017/S0013091500008221.

[5] Volkmar Felsch, Ludger Hippe, and Joachim Neubüser. *ITC – a GAP package, Version 1.4.* 2004. URL: http://www.gap-system.org/Packages/itc.html.

[6] Greg Gamble et al. *ACE – a GAP package, Version 5.1.* 2012. URL: http://www.gap-system.org/Packages/ace.html.

[7] *GAP – Groups, Algorithms, and Programming, Version 4.6.5.* The GAP Group. 2013. URL: http://www.gap-system.org.

[8] Derek F. Holt, Bettina Eick, and Eamonn A. O'Brien. *Handbook of computational group theory.* Discrete Mathematics and its Applications (Boca Raton). Chapman & Hall/CRC, Boca Raton, FL, 2005, pp. xvi+514. URL: http://dx.doi.org/10.1201/9781420035216.

[9] Joachim Neubüser. “An elementary introduction to coset table methods in computational group theory”. In: [2], pp. 1–45. URL: http://dx.doi.org/10.1017/CBO9780511661884.004.