An automated proof theory approach to computation with permutation groups

Arjeh M. Cohen
Department of Mathematics
Eindhoven University of Technology
Netherlands

Scott H. Murray
School of Mathematics and Statistics
University of Sydney
Australia

October 21, 2002

Abstract

This is an introduction to data structures for permutation groups with a proof theoretic flavour.

1 Introduction

In this paper, we introduce the reader to computational permutation group theory. We describe the basic concepts and first results in this area of mathematics, as well as the data structures required to do actual computations. We follow [1]. Although our aim is to apply the theory to the graph isomorphism problem, in the present notes we only go as far as determining the order of a group generated by a given set of permutations.

In the spirit of the Calculemus Autumn School, our approach will be proof theoretic. Computer algebra has always had an emphasis on complexity of algorithms, so that bigger and bigger problems could be solved on a given machine. The internet will play an increasingly large role in the exchange of mathematics between people, and we believe this will require a different approach to computational mathematics. As the exchange of mathematics across the World Wide Web becomes easier than solving all

problems locally, the management of mathematical queries becomes more prominent. The problem of verifying the correctness of computations is particularly acute when they are no longer done on local machines with software the user trusts.

In this paper, our *queries* are invocations of permutation group algorithms that have been developed over the years and are implemented as part of the GAP computer algebra package. The *response* to a query is the output of the algorithm, which may have been run on a remote computer which the user knows nothing about. The user has reason to doubt the validity of the response, and so will demand some kind of *verification*. Since our queries are mathematical in nature, this verification should take the form of (an encoding of) a proof.

A classical example is the factorization of a natural number. If a sequence $p_1, p_2, \ldots, p_t$ of numbers is returned as a response to the query "factor the natural number n," it is easy for the user to verify whether $n = p_1 \cdots p_t$. In order to verify that each p_i is a prime number, it would be very useful to receive additional data, such as the primality witnesses for each p_i. This example has been worked out by Olga Caprotti, Martijn Oostdijk and the first author [3, 2].

We treat computational permutation group theory in a similar manner. We give additional data that allows for a relatively easy check of the correctness of the answer. Of course this requirement may prevent us from using the most efficient possible algorithms and implementations. In general, we use functions in the computer algebra system GAP, which are close to the state of the art. However, on occasion we have been forced to implement simpler methods that allow us to provide the data for a straightforward verification of the result.

This paper will be concerned with providing human readable proofs that could be transformed to a computer checkable proof without too much effort. In this way, we contribute to the integration of computer algebra and proof verification, which is the research focus of the Calculemus project.

2 Membership

We address the the question how to prove that the permutation g belongs to the group G. In computational permutation group theory, a group G is specified by a set of generating permutations A. Suppose that

$$A = \{a_1, a_2, \ldots, a_k\}$$

consists of permutations of the points $\Omega := \{1, 2, \ldots, n\}$, i.e., A is a subset of the symmetric group Sym_n. Hence, by the definition of a generating set, G is the unique smallest subgroup of Sym_n which contains A.

We define a *word* in A to be an expression of the form

$$a_{i_1}^{e_1} a_{i_2}^{e_2} \cdots a_{i_m}^{e_m}$$

where the indices i_j are in the range $1, \ldots, k$ and the exponents e_j are integers. It is now easily shown that the set of words in A form a subgroup of Sym_n, and it is obvious that there is no smaller subgroup containing A. Hence a permutation $g \in \mathrm{Sym}_n$ is an element of G if, and only if, it can be expressed as a word in A. (A note for those who know combinatorial group theory: since we make little use of words, we are not making the normal distinction between a word in the free group, and its evaluation in the symmetric group).

Writing an arbitrary permutation g as a word in A is a difficult computational problem, which is beyond the scope of this tutorial. Instead we just use the existing methods implemented in GAP without explaining how they work. See Section 8, for a technique to show that a permutation is *not* an element of G.

Example 2.1 The Mathieu group on 11 points, M_{11}, has generating set $A = \{a_1, a_2\}$, where:

$$\begin{aligned} a_1 &= (1,10)(2,8)(3,11)(5,7), \\ a_2 &= (1,4,7,6)(2,11,10,9). \end{aligned}$$

Query input

- A permutation g.
- A list A of permutations generating G.
- We are given the fact that $g \in \langle A \rangle$.

GAP input

```
G := Group(A);
IsIn_Proof(g,G);
```

GAP output A word in A that is equal to g.

Query output By definition, G is generated by A and so G consists of those elements that can be expressed as a word in A. In particular $g =$ `IsIn_Proof`(g, G), and so belongs to G.

3 Subgroup

Suppose H is another permutation group with generating set B. How to prove that H is a subgroup of G? From the definition of a generating set it follows that H is a subgroup of G if, and only if, every element of B is contained in G.

Query input

- A list A of permutations which generate G.
- A list B of permutations which generate H.
- We are given the fact that H is a subgroup of G.

GAP input

```
G := Group(A);
H := Group(B);
IsSubgroup_Proof(H,G);
```

GAP output A list of words in A indexed by B.

Query output In order to show that H is a subgroup of G, it suffices to show that each element of the generating set B belongs to G. `IsSubgroup_Proof`(H, G) presents, for each element b of B, how it can be expressed as a word in A. This establishes that each element of B belongs to G, and so H is a subgroup of G.

4 Orbit

The concept of an orbit is used in the construction of sets of coset representatives for stabilisers. Let G be a permutation group on $\{1, \ldots, n\}$. The *orbit* of x under the action of G is

$$xG = \{xg : g \in G\}.$$

Suppose G has generating set $A = \{a_1, a_2, \ldots, a_k\}$ and let X be an orbit of G. Then the orbit graph $\mathcal{G}$ (on X) has vertex set X, label set $\{1, 2, \ldots, k\}$ and labeled edges

$$y \xrightarrow{i} z \text{ where } ya_i = z.$$

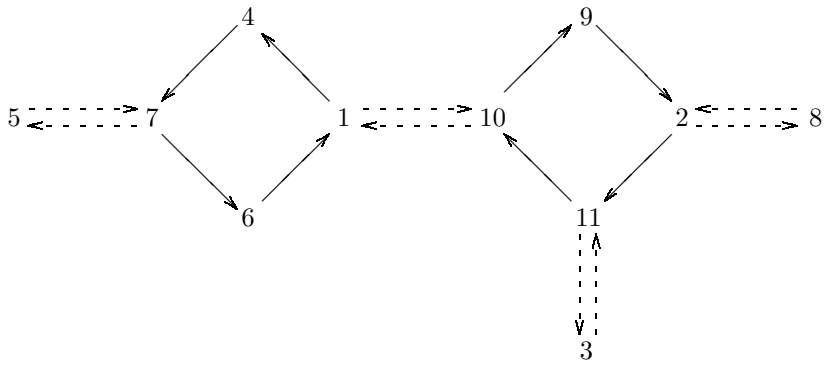

Figure 1: Orbit graph of M_{11}

Example 4.1 Consider $M_{11} = \langle s_1, s_2 \rangle$, where s_1 and s_2 are defined in Example 2.1. The action of $\{s_1, s_2\}$ on the orbit $X = \{1, 2, \ldots, 11\}$ is shown in Figure 1, where the dotted lines are labeled by 1 and solid lines by 2.

Algorithm 4.1 *Calculate orbit*

```
(* input:  x ∈ Ω, generating set A = {a_1, a_2, ..., a_k}.
   output: X = xG. *)
begin
    let X = {x};
    for y ∈ X, for a ∈ A do
        let y = ya;
        if y ∉ X then
            add y to X;
        end if;
    end for;
    return X;
end
```

This algorithm terminates after it has applied every generator to every point in X.

Query Input

- A list A of permutations generating G.
- A point x.

GAP input

```
G := Group(A);
Orbit_Proof(G,x);
```

GAP output A set of points X and a list B of words in A indexed by X. The set X is just the orbit xG. The word in A corresponding to $y \in X$ maps x to y.

Query output In order to show that X is the G-orbit of x, we need to show two statements:

1. Each element of A leaves the set X invariant. This is a straightforward check that the cycles containing points of X do not contain any points not in X.
2. Each element of X is image of x under an element of G. These elements are produced by the table `Orbit_Proof`(`Group`$(A), x$).

5 Schreier tree

Stabiliser subgroups are of fundamental importance to both theoretical and computational permutation group theory. The *stabiliser subgroup* in G of x is

$$G_x = \{g \in G : xg = x\}.$$

It is not immediately clear how to compute with this subgroup, since the definition gives us a test for whether g is an element of G_x, but does not give us for example a generating set.

The following lemma gives us a one-to-one correspondence between the orbit of a point and the set of cosets of its stabiliser.

Lemma 5.1 (Orbit Lemma) *If $y \in xG$, then $\{g \in G : xg = y\}$ is a coset of G_x. In particular, $|xG| = |G|/|G_x|$.*

Proof: Choose $h \in G$ such that $xh = y$; then

$$\begin{aligned} \{g \in G : xg = y\} &= \{g \in G : xg = xh\} \\ &= \left\{g \in G : xgh^{-1} = x\right\} \\ &= \left\{g \in G : gh^{-1} \in G_x\right\} = G_x h. \end{aligned}$$

Hence there is a one-to-one correspondence between the orbit and the cosets, and the second result follows. □

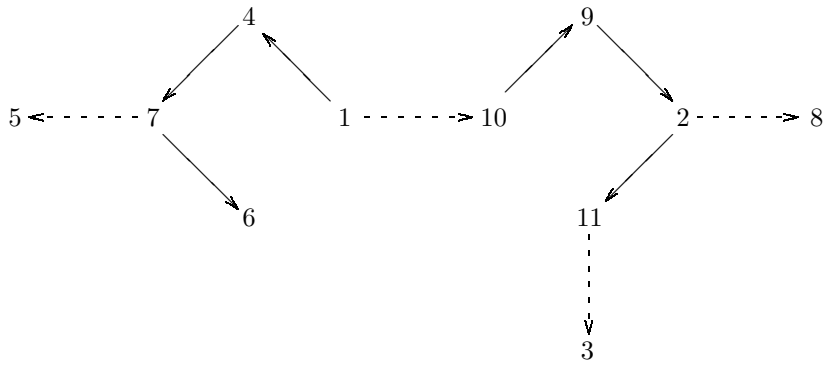

Figure 2: Schreier tree $\mathcal{T}$ for M_{11}

Suppose that for every element y of the orbit xG, we choose $t(y) \in G$ with the property that $xt(y) = y$. Then it follows immediately from the Orbit Lemma 5.1 that all such $t(y)$ form a set of coset representatives for G_x in G.

It would be inefficient to store all the elements $t(y)$, so instead we construct a *Schreier tree* rooted at x. That is, a subgraph $\mathcal{T}$ of the orbit graph $\mathcal{G}$ containing x which is a tree (when we view the edges as being undirected) with root x. For every $y \in X$, there is a unique minimal path in $\mathcal{T}$ from x to y (again, disregarding the fact that the labeled edges are directed).

Example 5.1 Consider M_{11} with the generating set of Example 2.1. Then $\mathcal{G}$ is the graph shown in Figure 1, where dotted lines are labeled by 1 and solid lines by 2. A Schreier tree $\mathcal{T}$ rooted at 1 is shown in Figure 2.

In practice, we store this tree in a linearized form, using two vectors $v : X \to \{-m, \ldots, -1, 0, 1, \ldots, m\}$ and $\omega : X \to X \cup \{0\}$ defined by:

$$\omega(z) = \begin{cases} y & \text{if } y \text{ is adjacent to } z \text{ and on the minimal path from } z \text{ to } x \\ 0 & \text{if } z = x \end{cases}$$

$$v(z) = \begin{cases} l & \text{if } y = \omega(z) \text{ and } y \xrightarrow{l} z \text{ is in } \mathcal{T} \\ -l & \text{if } y = \omega(z) \text{ and } y \xleftarrow{l} z \text{ is in } \mathcal{T} \\ 0 & \text{if } z = x \end{cases}$$

We call v the *Schreier vector* and ω the *backpointers*. They can be computed by a modified version of the orbit algorithm.

Algorithm 5.1 *Calculate orbit*

```
(* input:  x ∈ Ω, generating set A = {a_1, a_2, ..., a_k}.
   output: X = xG, Schreier vector v, vector of backward pointers ω. *)
begin
    let X = {x};
    let v(x) = ω(x) = 0;
    for x ∈ X, for j = 1, ..., k do
        let y = x a_j;
        if y ∉ X then
            add y to X;
            let v(y) = j, ω(y) = x;
        end if;
        let y = x a_j^{-1};
        if y ∉ X then
            add y to X;
            let v(y) = -j, ω(y) = x;
        end if;
    end for;
    return X, v, ω;
end
```

Example 5.2 The linearized version of the Schreier tree in Figure 2, rooted at 1, is given in the following table.

	1	2	3	4	5	6	7	8	9	10	11
v	0	2	1	2	1	2	2	1	2	1	2
ω	0	9	11	1	7	7	4	2	10	1	2

We can also choose the root in Figure 2 to be 5. This results in the following linearized version of the Schreier tree rooted at 5.

	1	2	3	4	5	6	7	8	9	10	11
v	-2	2	1	-2	0	2	-1	1	-2	1	2
ω	4	9	11	7	0	7	5	2	10	1	2

Query Input

- A list $A = [a_1, \ldots, a_k]$ of permutations generating G.
- A point x.

GAP input

```
G := Group(A);
SchreierData(G,x);
```

GAP output A triple $[X, v, \omega]$ of integer sequences consisting of the orbit, the Schreier vector, and the backpointers.

Query output Consider the table `SchreierData`(G, x). It has three rows, the first of which represents the orbit $X = xG$. In order to show that X is indeed the G-orbit containing x, see Section 4. To show that v is a Schreier vector and ω its backpointers, consider a column of the table, say x_j, v_j, ω_j. It suffices to show that $x_j a_{-v_j} = \omega_j$ if $v_j < 0$ and that $x_j a_{v_j}^{-1} = \omega_j$ if $v_j > 0$. This is a (perhaps tedious but) trivial check.

The Schreier tree enables us to create a set U of coset representatives for G_x in G and a map $t : G \to U$ sending an element g of G to the representative of $G_x g$ as follows. For $g \in G$, there is a minimal path $x = x_0, x_1, \ldots, x_m = xg$ in the Schreier tree $\mathcal{T}$ from x to xg. Write $b_j = a_{v(x_j)}$ if $v(x_j) > 0$ and $b_j = a_{-v(x_j)}^{-1}$ if $v(x_j) < 0$. Then $t(g) = b_1 \cdots b_m$ satisfies $xt(g) = xg$ and $t(g) = t(hg)$ whenever $h \in G_x$. So, taking U to be the image of the map $t : G \to G$, we find that U and t are as required.

Example 5.3 The values of t depend only on the coset of G_x in which the argument lies. Hence we can also give t as a function on the orbit X of G with the understanding that $t(g) = t(xg)$.

For the group M_{11} with generators as introduced in Example 2.1, with the Schreier tree rooted at 1 as discussed in Example 5.2, the values of t on $X = \{1, \ldots, 11\}$, written as words in $A = [a_1, a_2]$, are as follows.

y	1	2	3	4	5	6	7	8	9	10	11
$t(y)$	1	$a_1 a_2^2$	$a_1 a_2^{-1} a_1$	a_2	$a_2^2 a_1$	a_2^{-1}	a_2^2	$a_1 a_2^2 a_1$	$a_1 a_2$	a_1	$a_1 a_2^{-1}$

As suggested by the example, the set U of values of t can be used to construct the table in part 2 of the proof in the end of Section 4 that X is a G-orbit.

6 Stabiliser

Now that we have a set of coset representatives for G_x, we can use it to compute a generating set, with the following lemma.

Lemma 6.1 (Schreier's lemma) *Suppose G is a group with generating set A, and H is a subgroup of G. If U is a set of coset representatives for H in G, and the function $t : G \to U$ maps an element g of G to the representative of Hg, then a generating set for H is given by*

$$\left\{ua\,t(ua)^{-1} : u \in U, a \in A\right\}.$$

Proof: Throughout the proof we shall use the fact that, for $g, h \in G$, we have $t(g) = t(h)$ if and only if $Hg = Hh$.

Let h be an element of H. It can be written in the form $b_1 b_2 \cdots b_l$, where each b_i, or its inverse, is in A. Let $u_i = t(b_1 b_2 \cdots b_i)$, so that $Hb_1 b_2 \cdots b_i = Hu_i$ for $i = 0, 1, \ldots, l$. Then $u_0 = t(1) = 1$ and $u_l = t(h) = 1$, so

$$h = u_0 h u_l^{-1} = (u_0 b_1 u_1^{-1})(u_1 b_2 u_2^{-1}) \cdots (u_{l-1} b_l u_l^{-1}).$$

Therefore, it suffices to show each $u_{i-1} b_i u_i^{-1}$ is of the form $\left(ua\,t(ua)^{-1}\right)^{\epsilon}$ where $u \in U, a \in A, \epsilon \in \{\pm 1\}$. Consider $u_{i-1} b_i u_i^{-1}$, for $i = 1, 2, \ldots, l$. Now $u_i = t(b_1 b_2 \cdots b_i) = t(u_{i-1} b_i)$, since $Hb_1 b_2 \cdots b_i = Hu_{i-1} b_i$. Let $u = u_{i-1} \in U$ and $b = b_i$; we can now write

$$u_{i-1} b_i u_i^{-1} = ub\,t(ub)^{-1}.$$

This has the desired form if $b \in A$ (with $\epsilon = 1$); otherwise, $b = a^{-1}$ for some $a \in A$ and let $v = t(ua^{-1}) \in U$. Since $Hva = Hua^{-1}a = Hu$, we have $t(va) = u$, and so the inverse of $ub\,t(ub)^{-1}$ can be written

$$t(ua^{-1})\,au^{-1} = va\,t(va)^{-1},$$

which has the desired form (with $\epsilon = -1$). The result now follows. □

Observe that for $x \in \Omega$ and $H = G_x$, the function $t : G \to U$ does not depend on the choice of element in a coset Hg, so the map $t : \Omega \to G$ given by $t(xg) = t(g)$, is well defined. This indicates how to apply Schreier's lemma 6.1 to permutation groups.

Query Input

- A list A of permutations generating G.
- A point x.

GAP input

```
G := Group(A);
H := Stabilizer(G,x);
IsStabiliser_Proof(G,x,H);
```

GAP output

- A proof that H is a subgroup of G (see Section 3).
- A triple $[X, v, \omega]$ of integer sequences consisting of the orbit, the Schreier vector, and the backpointers.
- A sequence of quadruples (y, i, g, h) consisting of a point $y \in X$, an index $i = 1, \ldots, |A|$, the Schreier generator $t(y)a_i t(t(y)a_i)^{-1}$, where a_i is the i-th element of A, written as a word g in the generators A of G, and as a word h in the generators of H.

Query output As in Section 3 we show that H is a subgroup of G. As in Section 5 we determine the Schreier tree $[X, v, \omega]$ for G at x. From the Schreier tree we find that the Schreier elements $t(y)$ for $y \in X$. We check that each Schreier generator is in H. By Schreier's lemma 6.1, we conclude that H is the stabiliser in G of x.

7 Base

Now that we have a stabiliser of a subgroup, we can repeat the process to form a chain of subgroups. A *base* for G is a finite sequence $B = [x_1, \ldots, x_k]$ of distinct points in Ω such that

$$G_{x_1, x_2, \ldots, x_k} = 1.$$

Hence, the only element of G which fixes all of the points $x_1, x_2, \ldots, x_k$ is the identity. Clearly every permutation group has a base, but not all bases for a given group are of the same length. If we write $G^{(i)} = G_{x_1, x_2, \ldots, x_i}$, then we have a *chain of stabilisers*

$$G = G^{(0)} \geq G^{(1)} \geq \cdots \geq G^{(k-1)} \geq G^{(k)} = 1.$$

We often require that a base has the additional property that $G^{(i)} \neq G^{(i+1)}$.

A base can be constructed by starting with $B = [x_1]$, and recursively choosing a letter x_i in a nontrivial $G_{x_1, \ldots, x_{i-1}}$-orbit and appending it to B. The construction is finished when $G_{x_1, \ldots, x_i} = 1$.

Example 7.1 The Mathieu group on 11 points, M_{11}, has a base $[1, 2, 3, 4]$. The Schreier trees and backpointers for this stabiliser chain are given in the following table

	1	2	3	4	5	6	7	8	9	10	11
v_1	0	2	1	2	1	2	2	1	2	1	2
ω_1	0	9	11	1	7	7	4	2	10	1	2
v_2	–	0	3	7	4	7	4	6	4	6	5
ω_2	–	0	2	5	3	11	5	11	7	4	5
v_3	–	–	0	7	4	7	4	6	4	6	5
ω_3	–	–	0	5	3	11	5	11	7	4	5
v_4	–	–	–	0	7	5	6	6	7	6	7
ω_4	–	–	–	0	6	4	6	11	10	4	4

Query Input A list A of permutations generating G.

GAP input

```
G := Group(A);
B := BaseOfGroup(G);
IsBase_Proof(G,B);
```

GAP output A base $B = [x_1, \ldots, x_k]$ and, for each $i = 1, \ldots, k$, a set A_i of generators of the stabiliser of x_i in $\langle A_{i-1} \rangle$.

Query output Let l be the length of B. From `IsBase_Proof(G,B)` we read off, for $i = 1, \ldots, k$,

- a sequence A_i of permutations;
- a proof that $\langle A_i \rangle$ is the stabiliser of x_i in $\langle A_{i-1} \rangle$;
- the determination of the $\langle A_{i-1} \rangle$-orbit of x_i.

Having checked that $A_k = 1$, we conclude that B is a base with stabiliser chain the groups $\langle A_i \rangle$ for $i = 1, \ldots, k$.

8 Nonmembership

Here we deal with the complementary problem to the first one treated: Prove that the permutation g does not belong to G. Given a base $B = [x_1, \ldots, x_k]$ of G, we have a chain of subgroups

$$G = G^{(0)} \geq G^{(1)} \geq \cdots \geq G^{(k-1)} \geq G^{(k)} = 1$$

and sets $U^{(i)}$ consisting of coset representatives for $G^{(i+1)}$ in $G^{(i)}$. For we can take $G^{(i)} = G_{x_1,\ldots,x_{i-1}}$ and $U^{(i)} = t(G^{(i)})$ the set of Schreier elements corresponding to a Schreier tree for $G^{(i)}$ rooted at x_{i-1}.

An element g of G is contained in exactly one coset of $G^{(1)}$ in $G^{(0)}$, so $g = h_1 u_0$ for some unique h_1 in $G^{(1)}$ and u_0 in $U^{(0)}$. By induction, we can show that

$$g = u_k u_{k-1} \cdots u_1 u_0$$

where each $u_i \in U^{(i)}$ is uniquely determined by g. This process, called *sifting* an element, gives a canonical form for the elements of G and underpins most of the more advanced applications of stabiliser chains.

On the other hand, if g is not in G, then sifting fails because at some stage we get that $x_i h_{i-1}$ is not in the orbit $x_i G^{(i-1)}$, and so h_{i-1} is not in $G^{(i-1)}$. This gives us our proof of nonmembership.

Query Input

- A list A of permutations generating G.
- A permutation g.
- The fact that g is not in G.

GAP input

```
G := Group(A);
IsNotIn_Proof(g,G)
```

GAP output

- A base $B = [x_1, \ldots, x_k]$ together with generators for the stabiliser subgroups $G^{(i)} = G_{x_1,\ldots,x_i}$, the corresponding Schreier trees for $G^{(i-1)}$ rooted at x_i, and Schreier generators A_i for each base point x_i (so $G^{(i)} = \langle A_i \rangle$).
- A sequence of permutations $[h_0, h_1, \ldots, h_{j-1}]$ (where $j \leq k$) such that $h_i \in G^{(i)}$ and $x_i h_{i-1} \cdots h_1 h_0 = x_i g$ for $i = 1, \ldots, j-1$ and $x_j h_{j-1} \cdots h_1 h_0 \notin x_j G^{(j-1)}$.

Query output A proof that the base B and the corresponding stabiliser chain is correct is given in Section 7. Let $h = h_{j-1} \cdots h_0$. Then, as can be straightforwardly checked, h is in G and gh^{-1} fixes $x_1, \ldots, x_{j-1}$. But $x_j g h^{-1} \notin x_j G^{(j-1)}$, so gh^{-1} does not belong to $G^{(j-1)}$ whence not to G. Since $h \in G$, it follows that g does not belong to G.

At the cost of engineering with B, we can be a little more efficient. For, a closer look at the proof shows that we do not need the full base B, but only the first j elements.

9 Order

The order of a permutation group can now be effectively computed.

Lemma 9.1 (Order lemma) *Suppose G is a permutation group and $B = [x_1, \ldots, x_k]$ is a base for G. Then*

$$|G| \;=\; \prod_{i=1}^{k} |x_i G^{(i-1)}|$$

Proof: Follows directly from repeated application of the Orbit lemma 5.1:

$$\begin{aligned} |G| &= |x_1 G| \cdot |G^{(1)}| \\ &= |x_1 G| \cdot |x_2 G^{(1)}| \cdot |G^{(2)}| \\ &= \cdots \end{aligned}$$

as $G^{(k)}$ is trivial. □

Example 9.1 From Example 7.1 it immediately follows that the Mathieu group on 11 points has order

$$|1\, M_{11}| \cdot |2\, M_{11}^{(1)}| \cdot |3\, M_{11}^{(2)}| \cdot |4\, M_{11}^{(3)}| = 11 \cdot 10 \cdot 9 \cdot 8 = 7920.$$

Query Input A list A of permutations generating G.

GAP input

```
G := Group(A);
Order_Proof(G);
```

GAP output A base $B = [x_1, \ldots, x_k]$, the corresponding stabiliser chain $G^{(i)}$, and the sizes of the orbits $x_i G^{(i-1)}$.

Query output The proof that the base and the stabiliser chain are correct is given in Section 7. By the Order lemma 9.1, the order of G is the product of the orbit sizes $|x_i G^{(i-1)}|$ for $i = 1, \ldots, k$.

10 Exercises

The stars in front of an exercise indicate the level of difficulty. The more stars, the more difficult the exercise. If you solve a * exercise, that's fine, you're beginning to understand the material. If you solve a ** exercise and you haven't studied the permutation groups before, that's something to be proud of. If you solve a *** exercise, tell me about it. If you solve a **** exercise, I am really interested in your solution. If you solve a ***** exercise, you are paving the road for the sequel to these notes (on graph automorphism groups and graph isomorphisms).

Exercise 10.1 *. Let $\Gamma = (\Omega, E)$ be a graph. (Say, without loops, multiple edges, undirected.) A permutation g of Ω is called an *automorphism* of Γ is it preserves E. This means that E, viewed as a subset of Y_2 (see Exercise 10.11) is a union of G-orbits. Show that the set of all automorphisms of Γ is a permutation group.

Exercise 10.2 *. Write a proof generator in GAP for

$$\langle A \rangle = \langle B \rangle$$

where A and B are lists of permutations. (So, you asked to write an algorithm that will provide a proof given the fact that the identity holds for inputs A, B.)

Exercise 10.3 *. Similarly for

$$\langle A \rangle \neq \langle B \rangle.$$

Exercise 10.4 **. A subgroup N of a group G is called *normal* if $gNg^{-1} = N$ for all $g \in G$. Write a proof generator in GAP for

$$\langle B \rangle \text{ is a normal subgroup of } \langle A \rangle$$

where A and B are lists of permutations.

Exercise 10.5 **. Similarly for

$$\langle B \rangle \text{ is not a normal subgroup of } \langle A \rangle.$$

Exercise 10.6 *. Let G be a finite group and let p be a prime. A subgroup S of G is called a Sylow p-subgroup if the order $|S|$ of S is a power of p and

the index of S in G, that is $|S\backslash G|$, is not divisible by p. Such subgroups always exist. Write a proof generator in GAP for

$$\langle B\rangle \text{ is a Sylow } p\text{-subgroup of } \langle B\rangle$$

where A and B are lists of permutations.

Exercise 10.7 *. Two elements $g, h \in G$ are called *conjugate* if there is an element x of G such that $xgx^{-1} = h$; notation $g \sim h$. Write a proof generator in GAP for $g \sim h$.

Exercise 10.8 *. Let G, H be groups. A homomorphism $\varphi : G \to H$ is a map satisfying $\varphi(gh) = \varphi(g)\varphi(h)$ for all $g, h \in H$. A *permutation representation* of a group G is a homomorphism $G \to \mathrm{Sym}(X)$ for some set X.

Show that each group G with subgroup H has a permutation representation $\varphi_H : G \to \mathrm{Sym}(H\backslash G)$ given by $\varphi_H(g) = Hg$.

Exercise 10.9 *. Let Ω, Ξ be two sets. Two permutation representations $\varphi : G \to \mathrm{Sym}(\Omega)$ and $\psi : G \to \mathrm{Sym}(\Xi)$ are called *equivalent* if there is a bijective map $T : \Omega \to \Xi$ such that, for each $g \in G$, we have $\varphi(g)T = T\psi(g)$ (writing T as a map that acts from the right, just like $\varphi(g)$ and $\psi(g)$). Prove that the name is justified: that equivalence is an equivalence relation on the collection of permutation representations.

Equivalent representations can be considered 'the same up to renaming elements.'

Exercise 10.10 **. A permutation representation $\varphi : G \to \mathrm{Sym}(\Omega)$ is called *transitive* if Ω is a single $\varphi(G)$-orbit.

Let H be a subgroup of G. Prove that the permutation representation φ_H of Exercise 10.8 is transitive.

Conversely, show that each transitive permutation representation of G is equivalent to one of the form φ_H for a subgroup H of G. (Hint: pick $x \in \Omega$ and take $H = G_x$.)

Exercise 10.11 *. Let G be a permutation group on $\Omega = \{1, \ldots, n\}$. Denote by Y the collection of subsets of Ω. For $x = \{x_1, \ldots, x_k\} \in Y$, write

$$x\varphi(g) = \{x_1 g, \ldots, x_k g\}.$$

Show that the resulting map $\varphi : G \to \mathrm{Sym}(Y)$ is a permutation representation.

Now, for $k \in \{1, \ldots, n\}$, show that we can restrict φ to $\varphi_k : G \to \mathrm{Sym}(Y_k)$, where Y_k is the collection of subsets of Ω of size k. (Hint: this means that Y_k is a union of $\varphi(G)$-orbits.)

Exercise 10.12 ***. Let $\varphi : G \to \mathrm{Sym}(X)$ be a permutation representation. Recall that $\ker \varphi = \{g \in G \mid \varphi(g) = 1\}$. Give an algorithm to find generators for $\ker \varphi$ if G is generated by a finite list A of permutations. (Hint: deal with stabilisers in G, Schreier elements, and so on, in this greater generality, and identify $\ker \varphi$ as the set of elements of G fixing a base of $\mathrm{Im}\, \varphi$.)

Exercise 10.13 *. Let G be a permutation group on Ω. A subset B of Ω is called a *block* if $gB \cap B = B$ or $\emptyset$ for each $g \in G$.

Prove that if B is a block and G is transitive, we have a partition $\{B = B_1, \ldots, B_k\}$ of Ω which is preserved by G (that is, for each $g \in G$ and $i \in \{1, \ldots, k\}$, there exists j such that $gB_i = B_j$.)

Exercise 10.14 **. Let G be a permutation group on Ω. We say that G is *primitive* if the only blocks of G on Ω (see Exercise 10.13) are of size 1 or $|\Omega|$. Give a proof generator in GAP for

$$\langle A \rangle \text{ is not primitive}$$

where A is a finite list of permutations of Ω.

Exercise 10.15 ****. Let G be a permutation group on Ω. See Exercise 10.14 for the definition of primitivity. Give a proof generator in GAP for

$$\langle A \rangle \text{ is primitive}$$

where A is a finite list of permutations of Ω.

Exercise 10.16 *. Let G be a group. Its *center* is

$$\{x \in G \mid \forall g \in G\ gx = xg\}.$$

Prove that this is a normal subgroup (cf. Exercise 10.4) of G. Give an algorithm to find generators of the center of G in terms of generators of G. (Hint: apply Exercise 10.12 to the permutation representation $\kappa : G \to \mathrm{Sym}(G)$ given by $\kappa(g) : h \mapsto g^{-1}hg$ for $g, h \in G$.)

Exercise 10.17 **. Notation as in Exercise 10.7. Write a proof generator in GAP for $g \not\sim h$. (Hint: Show that g and h are not in the same $\kappa(G)$-orbit for κ as in Exercise 10.16.)

Exercise 10.18 **. Let $H = \langle B \rangle$ and $K = \langle C \rangle$ be subgroups of $G = \langle A \rangle$. Give an algorithm to determine the size of the set

$$H\,K = \{hk \mid h \in H, k \in K\}.$$

(Hint: the answer equals $|H|\,m$, where m is the number of cosets Hk with $k \in K$. This is the size of the $\varphi_H(K)$-orbit of H, with φ_H as in Exercise 10.8.)

Use $|H\,K| \cdot |H \cap K| = |H| \cdot |K|$ to determine a generating set for the subgroup $H \cap K$ of G.

Exercise 10.19 *****. Write a proof generator in GAP for

$$\langle A \rangle \text{ is the automorphism group of } \Gamma$$

where $\Gamma = (\Omega, E)$ is a graph and A is a list of permutations of Ω.

Exercise 10.20 *. Write a proof generator in GAP for

$$\Gamma \text{ and } \Delta \text{ are isomorphic.}$$

Exercise 10.21 *****. Write a proof generator in GAP for

$$\Gamma \text{ and } \Delta \text{ are not isomorphic.}$$

References

[1] Cuypers, Soicher, and Sterk: Working with finite groups, in "Some Tapas of Computer Algebra" (eds. A.M. Cohen, H. Cuypers, H. Sterk), Springer, Heidelberg, 1999.

[2] A.M. Cohen, Communicating Mathematics across the Web, pp. 283–300 in "Mathematics Unlimited – 2001 and beyond" (eds. Björn Engquist and Wilfried Schmid) Springer, Heidelberg, 2000.

[3] Caprotti & Oostdijk: Pocklington.