Programming in GAP: Working on problems from scratch

GAPDays Summer 2025

Meike Weiss and Lukas Schnelle

August 2025

Groups in GAP can be used in a number of ways. We will start by creating a permutation group:

```
gap> g:=Group((1,2,3), (3,4));
Group([ (1,2,3), (3,4) ])
```

With this group can can ask e.g. the order of the group:

```
gap> Size(g);
24
```

or create subgroups:

```
gap> h:=Subgroup(g, [(1,2,3)]);
Group([ (1,2,3) ])
```

Most other Group properties can also easily be queried:

```
gap> IsNilpotent(g);
false
gap> IsSolvable(g);
true
gap> DerivedSubgroup(g);
Group([ (1,4,3), (1,2,3) ])
```

and of course the groups can operate on sets, e.g.:

```
gap> OnPoints(3, (1,2,3));
1
```

which computes $x^{-1}gx$ for `OnPoints(x, g)`.

With group actions, we also can compute the orbits of a group, acting on a set:

```
gap> Orbits(g, [1..5]);
[ [ 1, 4, 2, 3 ], [ 5 ] ]
```

as well as the stabilisers for a given element:

```
gap> Stabilizer(g, 1);
Group([ (3,4), (2,4,3) ])
```

There are also other presentations for groups. First we consider matrix groups.

```
gap> g:=Group( [[0,-1],[1,0]] );
Group([ [ [ 0, -1 ], [ 1, 0 ] ] ])
gap> Size(g);
4
```

If we choose a fitting starting vector to act on we can also translate a matrixgroup into a permutation group:

```
gap> v:=[1,0];
[ 1, 0 ]
gap> orb:=Orbit(g, v, OnPoints);
[ [ 1, 0 ], [ 0, -1 ], [ -1, 0 ], [ 0, 1 ] ]
gap> hom:=ActionHomomorphism(g, orb, OnPoints);
<action homomorphism>
gap> p:=Image(hom);
Group([ (1,2,3,4) ])
```

We can also check if the two groups are isomorphic by constructing an isomorphism:

```
gap> iso:=IsomorphismGroups(p, g);
[ (1,2,3,4) ] -> [ [ [ 0, -1 ], [ 1, 0 ] ] ]
```

If they are not isomorphic the function would return `fail`.

Definition

A group G is called **polycyclic** if there is a chain of normal subgroups such that

$$G \trianglerighteq G_n \trianglerighteq G_{n-1} \trianglerighteq \cdots \trianglerighteq G_0 = 1,$$

with 1 being the trivial group and for all $1 \leq i \leq n$ the factors C_i/C_{i-1} are cyclic.

PcGroups are often used in GAP as they often provide efficient computations. Thus, several functions in GAP return these kinds of groups by default:

```
g:=AbelianGroup([2,3]);
<pc group of size 6 with 2 generators>
```

Definition

A group G is called **finitely presented** if there exists a free group F with generating set $S = \{a_1, \dots a_n\}$ and a set $R = \{r_1, \dots r_k\}$ of words in S so that

$$G \cong F/\langle r_1, \dots r_k \rangle.$$

The set R is called the **relators**. Often the group is denoted by $\langle a_1, \dots a_n \mid r_1 = \cdots = r_k = 1 \rangle$.

These groups are often easy to construct, but checking if the resulting group is finite is an open problem.

Constructing finitely presented groups is straight forward:

```
gap> f:=FreeGroup("a", "b");
<free group on the generators [ a, b ]>
gap> g:=f/[f.1^4, f.2^2, (f.1*f.2)^2,
f.1*f.2*f.1^(-1)*f.2^(-1)*f.1^(-2)];
<fp group on the generators [ a, b ]>
gap> RelatorsOfFpGroup(g);
[ a^4, b^2, (a*b)^2 , a*b*a^-1*b^-1*a^-2]
gap> Size(g);
8
```

As the presentation is not unique there are often nicer presentations. One option to compute these is applying Tietze transformations, which change the presentation isomorphically and see if the resulting presentation is "nicer".

```
gap> p:=PresentationFpGroup(g);
<presentation with 2 gens and 4 rels of total length 14>
gap> TzGoGo(p);
#I there are 2 generators and 3 relators of total length 10
gap> RelatorsOfFpGroup(FpGroupPresentation(p));
[ b^2, a^4, (a*b)^2 ]
```

Polycylic groups can also be converted into finitely presented groups

```
gap> g:=AbelianGroup([2,3]);
<pc group of size 6 with 2 generators>
gap> iso:=IsomorphismFpGroup(g);
[ f1, f2 ] -> [ F1, F2 ]
gap> h:=Image(iso);
<fp group of size 6 on the generators [ F1, F2 ]>
gap> RelatorsOfFpGroup(h);
[ F1^2, F2^-1*F1^-1*F2*F1, F2^3 ]
```

One very powerful function is `StructureDescription`. It tries to identify the given group and give a "nice" description.

```
gap> g:=AbelianGroup([0,2,3]);
<fp group of size infinity on the generators [ f1, f2, f3 ]>
gap> StructureDescription(g);
"Z x C6"
```

Another very useful functionality is the **small groups library**. It allows you to search a large number of known groups and filter them by properties you define yourself.

```
gap> grps:=AllSmallGroups(Size, [10..50], DerivedLength,
[3..5], IsSolvableGroup);
[ <pc group of size 24 with 4 generators>,
<pc group of size 24 with 4 generators>,
<pc group of size 48 with 5 generators>,
<pc group of size 48 with 5 generators>,
<pc group of size 48 with 5 generators>,
<pc group of size 48 with 5 generators>,
<pc group of size 48 with 5 generators>,
<pc group of size 48 with 5 generators> ]
```

Construct a permutation group with an action on tuples

Consider the group $GL(4, 53)$ as a permutation group and compute its order.

Consider the following group:

$$G := \langle a, b, c, d, e, f \mid a^2, d^3, e^3, b * e * b^{-1} * e, b^4, e^{-1} * a * e * a, b^{-1} * d * b * d,$$
$$b^{-1} * a * b * a, (a * f)^2, f^{-1} * b^{-1} * f * b, d^{-1} * a * d * a,$$
$$f^{-1} * d^{-1} * f * d, f^{-1} * e^{-1} * f * e, f^5, (e^{-1} * d)^3, (d^{-1} * e^{-1})^3 \rangle$$

This group is also available in the github repository under
`Exercises2/mysterygroup.g`:
`https://github.com/MeikeWeiss/GAP-Days2025-Intro`
Test the following properties:

- The order of the group
- If the group is nilpotent
- If the group is solvable
- The derived series and its factors
- Is this group polycyclic?
- Is the group elementary abelian?

Can you completely characterise the group?

More exercises from last year can be found here:
https://www.ilariacolazzo.info/gap/tutorials/sheet2/.