RAMP Demo

If you are used to working with regular or chiral polytopes / maniplexes in terms of their groups, then RAMP makes it easy to input such objects, and then to extract a variety of combinatorial and algebraic information. The following demo will guide you through some key features.

Regular polytopes and maniplexes

Defining regular polytopes and maniplexes

The simplest way to create a regular polytope is to describe its Schlafli symbol, and the extra relations that are needed to define the group beyond the universal ones. For example:

```
gap> P := AbstractRegularPolytope([4,3,4], "(r0 r1 r2 r3)^6");
AbstractRegularPolytope([ 4, 3, 4 ], "(r0 r1 r2 r3)^6")
gap> Size(P);
1536
```

So we have created the polytope of type $\{4,3,4\}$ whose automorphism group is the quotient of $[4,3,4]$ by the single extra relation $(\rho_0\rho_1\rho_2\rho_3)^6 = 1$. In RAMP, we use r instead of ρ.

If you execute this, you will notice that there is a slight lag. That is because RAMP is verifying that the input really defines a polytope. Consider:

```
gap> P := AbstractRegularPolytope([4,3,4], "(r0 r1 r2 r3)^4");
Error, The given input does not define a polytope. at /opt/gap-
4.12.1/.libs/../pkg/RAMP/lib/utils.gi:47 called from...
```

If you want to suppress this check, you can add NC at the end. (This is a common GAP idiom; NC stands for "no check".)

```
 P := AbstractRegularPolytopeNC([4,3,4], "(r0 r1 r2 r3)^4");
AbstractRegularPolytope([ 4, 3, 4 ], "(r0 r1 r2 r3)^4")
```

Naturally, if you use this and the resulting output is just a maniplex (not a polytope), then some functions may give undefined results. If you are not sure if the result will be a polytope, just build a maniplex instead, in essentially the same way.

```
gap> P := ReflexibleManiplex([4,3,4], "(r0 r1 r2 r3)^4");
ReflexibleManiplex([ 4, 3, 4 ], "(r0 r1 r2 r3)^4")
```

(Note that we call these `reflexible' maniplexes. I'll add some synonyms later so that you can also call them regular maniplexes.)

Under the hood, everything is a maniplex - a polytope is (usually) just a maniplex that satisfies "IsPolytopal". There are a few concepts that only really make sense for polytopes (such as tightness), but most routines work with both polytopes and maniplexes.

These constructors all have some abbreviations: `ARP` and `ARPNC` for making polytopes, `RefMan` for making maniplexes.

Combinatorial and algebraic information

Once you have a regular polytope or maniplex, you can easily get all kinds of information about it.

```
gap> P := AbstractRegularPolytopeNC([4,3,4], "(r0 r1 r2 r3)^6");
AbstractRegularPolytope([ 4, 3, 4 ], "(r0 r1 r2 r3)^6")

gap> Fvector(P);
[ 32, 96, 96, 32 ]

gap> SchlafliSymbol(P);
[ 4, 3, 4 ]

gap> IsSelfDual(P);
true

gap> Facet(P);
regular 3-polytope of type [ 4, 3 ]

gap> Facet(P) = Cube(3);
true
```

Notice that we have defined equality of polytopes and maniplexes to be isomorphism. So, if you build something in two different ways, you can check whether they are isomorphic. You can also check whether polytopes cover one another. The order of arguments follows GAP's convention: `IsCover(P, Q)` returns true if Q is a cover of P.

```
gap> IsCover(ARP([2,2]), ARP([4,4]));
true
gap> IsCover(ARP([2,2]), ARP([4,3]));
false
```

Other representations of the automorphism group

By default, `AutomorphismGroup(P)` uses the representation of the automorphism group that was used to build P - e.g., in the examples above, it returns a finitely presented group. Sometimes we want a permutation group instead; many algorithms are more efficient if we have a permutation representation of the group. We can insist on getting a permutation group:

```
gap> AutomorphismGroup(P);
<fp group of size 1536 on the generators [ r0, r1, r2, r3 ]>

gap> g := AutomorphismGroupPermGroup(P);
<permutation group of size 1536 with 4 generators>

gap> NrMovedPoints(g);
32

gap> g.1;
(1,2)(3,4)(5,6)(7,9)(8,11)(10,13)(12,15)(14,18)(16,19)(17,22)(20,25)(21,24)(23,27)
(26,29)(28,31)(30,32)
```

We got the automorphism group as a permutation group of degree 32. It is also possible to specify that we want the automorphism group to act on various parts of P:

```
gap> g2 := AutomorphismGroupOnFlags(P);
<permutation group with 4 generators>
gap> NrMovedPoints(g2);
1536

gap> g3 := AutomorphismGroupOnEdges(P);
<permutation group with 4 generators>
gap> NrMovedPoints(g3);
96

gap> g4 := AutomorphismGroupOnChains(P, [0,3]);
<permutation group with 4 generators>
gap> NrMovedPoints(g4);
256
```

The last example has the automorphism group acting on chains of type $(0,3)$; i.e., sets of incident vertices with 3-faces.

Note that it is also possible to build a polytope from a permutation group, using the same commands `ARP` etc.

```
gap> P2 := ARP(g);
regular 4-polytope of type [ 4, 3, 4 ] with 1536 flags

gap> P2 = P;
true

gap> AutomorphismGroup(P2);
<permutation group of size 1536 with 4 generators>
```

Now, AutomorphismGroup returns the permutation group that we used to build `P2`. If we want a finitely presented group ("fpgroup" in GAP), then we can insist on that as well.

```
gap> h := AutomorphismGroupFpGroup(P2);
<fp group on the generators [ r0, r1, r2, r3 ]>

gap> RelatorsOfFpGroup(h);
[ r0^2, r1^2, r2^2, r3^2, (r0*r2)^2, (r0*r3)^2, (r1*r3)^2, (r0*r1)^4,
  r0*r1*r2*r0*r1*r0*r3*r2*r1*r3*r2*r3*r0*r1*r2*r0*r1*r0*r3*r2*r3*r1*r2*r3,
(r2*r1)^3, (r2*r3)^4 ]
```

If `g` is a finitely presented group, you should usually avoid calling `ARP(g)` or even `RefMan(g)`, because checking whether `g` is even an sggi can take a long time. Instead, use the technique described before, using the Schlafli symbol and the "extra" relations.

Constructions of regular polytopes

Many constructions of regular polytopes are built in to RAMP:

```
gap> Q := Cube(3);
Cube(3)

gap> Q2 := TrivialExtension(Q);
regular 4-polytope of type [ 4, 3, 2 ] with 96 flags

gap> Q3 := UniversalExtension(Q);
regular 4-polytope of type [ 4, 3, infinity ] with infinitely many flags

gap> Q3 = ARP([4,3,infinity]);
true

gap> M := Mix(Cube(3), CrossPolytope(3));
reflexible 3-maniplex
gap> IsPolytopal(M);
true
gap> SchlafliSymbol(M);
[ 12, 12 ]
gap> Size(M);
1152
gap> M;
regular 3-polytope of type [ 12, 12 ] with 1152 flags

gap> A := Amalgamate(Cube(3), CrossPolytope(3));
reflexible 4-maniplex of type [ 4, 3, 4 ]
gap> A = ARP([4,3,4]);
true

gap> T := ToroidalMap44([3,0]);
ToroidalMap44([ 3, 0 ])
gap> Tp := Petrial(T);
ReflexibleManiplex([ 6, 4 ], "((r0 r2)*r1*r2*r1)^3,(r0 r1 r2)^4")
gap> Tpd := Dual(Tp);
ReflexibleManiplex([ 4, 6 ], "(r2*(r0*r1)^2)^3,(r2*r1*r0)^4")
```

Non-regular polytopes and maniplexes

Where RAMP really shines is in its ability to work with maniplexes that are not regular alongside those that are regular, without having to fuss with the automorphism groups necessarily. Maybe the easiest way to get a non-regular maniplex is using various classic constructions.

```
gap> P := Pyramid(Cube(3));
Pyramid(Cube(3))

gap> Facets(P);
[ Pyramid(4), Cube(3) ]

gap> NumberOfFlagOrbits(P);
5

gap> SchlafliSymbol(P);
[ [ 3, 4 ], [ 3, 4 ], 3 ]
```

The "Schlafli symbol" here is generalized; it says that the 2-faces are triangles and squares, and that the sections of a 3-face over a vertex are also triangles and squares.

```
gap> STG := SymmetryTypeGraph(P);
Edge labeled graph with 5 vertices, and edge labels [ 0, 1, 2, 3 ]
gap> LabeledEdges(STG);
[ [ [ 1, 2 ], 0 ], [ [ 3 ], 0 ], [ [ 4 ], 0 ], [ [ 5 ], 0 ], [ [ 1 ], 1 ], [ [ 3 ],
1 ], [ [ 4 ], 1 ],
  [ [ 2, 5 ], 1 ], [ [ 1 ], 2 ], [ [ 2 ], 2 ], [ [ 3 ], 2 ], [ [ 4, 5 ], 2 ], [ [ 1
], 3 ], [ [ 2 ], 3 ],
  [ [ 3, 4 ], 3 ], [ [ 5 ], 3 ] ]
```

Many classic map operations are built in, including the Wythoffians.

```
gap> Q := Cube(3);
Cube(3)
gap> Q2 := Medial(Q);
Medial(Cube(3))
gap> NumberOfFlagOrbits(Q2);
2
gap> NumberOfVertexOrbits(Q2);
1
gap> NumberOfFacetOrbits(Q2);
2
```

You can ask for the automorphism group of a non-regular polytope, but it is usually not constrained to be in a particular form. Usually we work with the connection group instead.

```
gap> g := ConnectionGroup(Q2);
<permutation group of size 2304 with 3 generators>
gap> NrMovedPoints(g);
96
```

Recall that every non-regular polytope has a smallest regular (maniplex) cover whose automorphism group is isomorphic to the connection group.

```
gap> R := SmallestReflexibleCover(Q2);
reflexible 3-maniplex with 2304 flags
gap> SchlafliSymbol(R);
[ 12, 4 ]
gap> IsCover(Q2, R);
true
```

Note that when we ask for the automorphism group of a regular polytope as a finitely presented group, it will always be in the "standard" form:

```
gap> h := AutomorphismGroupFpGroup(R);
<fp group on the generators [ r0, r1, r2 ]>
gap> RelatorsOfFpGroup(h);
[ r0^2, r1^2, r2^2, (r0*r2)^2, ((r0*r1)^2*r0*r2*(r1*r0)^4*r2*r1)^2, (r1*r0)^12,
  (r1*r0)^3*r2*r1*r0*(r2*(r1*r0)^2*r1)^2*(r0*r1)^2*r2,
  r1*r2*(r1*r0)^3*((r1*r0)^2*r2)^2*r1*r0*(r1*r2)^2*(r0*r1)^2*r0*
((r1*r0)^2*r2)^2*r1*r0*r2, (r1*r2)^4 ]
```

Working with rotary maniplexes

It is possible to build rotary (chiral or regular) maniplexes from the "standard" version of their automorphism group in a way analogous to regular maniplexes.

```
gap> P := RotaryManiplex([4,4], "(s2^-1 s1) * (s2 s1^-1)^4");
RotaryManiplex([ 4, 4 ], "(s2^-1 s1) * (s2 s1^-1)^4")

gap> Size(P);
136

gap> IsChiral(P);
true

gap> IsPolytopal(P);
true

gap> Fvector(P);
[ 17, 34, 17 ]

gap> cg := ChiralityGroup(P);
```

```
Group(<fp, no generators known>)

gap> Size(cg);
17

gap> R := SmallestReflexibleCover(P);
reflexible 3-maniplex

gap> R = ToroidalMap44([17,0]);
true
```

By default, isomorphism of polytopes does not have to respect the choice of base flag. If you want isomorphism of rooted maniplexes, you can ask for that.

```
gap> T := ToroidalMap44([1,4]);
ToroidalMap44([ 1, 4 ])

gap> T2 := ToroidalMap44([4,1]);
ToroidalMap44([ 4, 1 ])

gap> T = T2;
true

gap> IsIsomorphicRootedManiplex(T,T2);
false

gap> IsIsomorphicRootedManiplex(T, EnantiomorphicForm(T2));
true
```

For rotary polytopes, there are commands RotationGroup, RotationGroupPermGroup, etc that are analogues of AutomorphismGroup etc. Naturally, if P is chiral, then `RotationGroup(P)` does the same thing as `AutomorphismGroup(P)`.

Most of the commands that we explored for regular maniplexes also work for rotary maniplexes.

Databases of maniplexes and polytopes

A number of data sets are incorporated into RAMP, coming from Marston Conder, Primoz Potocnik, and ourselves. For example, the following code loads up the regular polyhedra with up to 2000 flags, and then we can filter the list by various properties. (The double semicolon at the end just suppresses the output.)

```
gap> L := SmallRegularPolyhedra(2000);;
gap> Size(L);
7897
gap> L2 := Filtered(L, P -> not(IsDegenerate(P)));;
gap> Size(L2);
6900
gap> L3 := Filtered(L2, P -> IsSelfDual(P));;
```

```
gap> Size(L3);
242
gap> L4 := Filtered(L3, P -> IsOddInt(SchlafliSymbol(P)[1]));;
gap> Size(L4);
28
gap> L4[1];
Simplex(3)
gap> L4[2];
AbstractRegularPolytope([ 5, 5 ], "r1*r2*r0*(r1*r0*r2)^2 ")
```

If we set the size parameter too large, then we get a warning, but RAMP returns as much as it can.

```
gap> L := SmallRegularPolyhedra(4001);;
#I  The list of small regular polyhedra with more than 4000 flags is incomplete.
#I  The list of flat regular polyhedra with more than 4000 flags is incomplete.
```

For certain families of polyhedra, we build them on the fly rather than saving them to a file. Thus, even if we set the size parameter too large, we do get some polyhedra that are larger than that parameter.

```
gap> L := SmallRegularPolyhedra(4500);;
#I  The list of small regular polyhedra with more than 4000 flags is incomplete.
#I  The list of flat regular polyhedra with more than 4000 flags is incomplete.

gap> L2 := Filtered(L, P -> Size(P) > 4000);;
gap> Size(L2);
727

gap> L2[1];
AbstractRegularPolytope([ 2, 1001 ])

gap> L3 := Filtered(L2, P -> not(IsDegenerate(P)));;
gap> L3[1];
AbstractRegularPolytope([ 501, 4 ], "(r2*r1)^4*r0*(r1*r2)^4*r0,
(r2*r1)^2*r1^2*r0*r1*r2*r1*r0,(r2*r1)^3*(r1*r0)^2*r1*r2*(r1*r0)^2")
```

The polyhedra we build on the fly include degenerate polyhedra, toroidal polyhedra, and certain families of flat polyhedra.

There are also databases for chiral and regular maniplexes and polytopes: `SmallChiral3Maniplexes`, `SmallReflexible3Maniplexes`, etc.