Exploring Abstract Algebra with Computer Software
PREP Workshop 2004
Section 7: Group Homomorphisms

The command `GroupHomomorphismByImages(G,H,[list of generators of G], [list of images of these generators])` in GAP will create the specified homomorphism. For example:

```
gap> S3:= SymmetricGroup(3);
Sym([1..3])
gap> f1:= GroupHomomorphismByImages(S3,S3, [(1,2,3),(1,3)],
> [(1,3,2),(1,2)]);
[(1,2,3), (1,3)] -> [(1,3,2), (1,2)]
```

$f1$ is the homomorphism $f1 : S_3 \to S_3$ that maps (1,2,3) to (1,3,2) and (1,3) to (1,2).

```
gap> Image(f1, (2,3));
(2,3)
gap> Image(f1,(1,2));
(1,3)
```

The above tells us that $f1(2,3) = (1,3)$ and $f1(1,2) = (1,3)$ (as you can easily check).

```
gap> Size(Image(f1));
6
gap> Kernel(f1);
[()]
```

Thus $f1$ is an automorphism. (Again this is easy to check by hand.) For another example consider the following. Read through the GAP commands and be sure you understand the output.

```
gap> f2:= GroupHomomorphismByImages(S3,S3, [(1,2,3),(1,3)],
> [(),(1,2)]);
[(1,2,3), (1,3)] -> [ (), (1,2) ]
Size(Image(f2));
```

```
2
gap> H:=Image(f2);
Group([ (), (1,2) ])
gap> Image(f2, (2,3));
(1,2)
gap> Kernel(f2);
Group([(1,2,3)]);
```

If you define a map that is not a homomorphism, GAP will return `fail`

```
gap> f3:= GroupHomomorphismByImages(S3,S3, [(1,2,3),(1,3)],
[(1,3),(1,2)]);
fail
```

$f3$ maps (1,2,3) (an element of order 3) to (1,3) (an element of order 2) so $f3$ is not a homomorphism.

Recall the group D_n is a subgroup of S_n which is generated by a rotation of order n and a reflection. Thus $(1,2,3,\ldots,n)$ and $(1,n)(2,n-1)\ldots(\frac{n}{2},\frac{n}{2}+1)$ generate D_n when n is even and $(1,2,3,\ldots,n)$ and $(1,n)(2,n-1)\ldots(\frac{n-1}{2},\frac{n-1}{2}+2)$ generate D_n when n is odd. So, for example, every element in D_6 can be written as products of powers of $(1,2,3,4,5,6)$ and $(1,6)(2,5)(3,4)$ and every element in D_7 can be written as products of powers of $(1,2,3,4,5,6,7)$ and $(1,7)(2,6)(3,5)$.

One way to determine if a homomorphism from the finite group G to itself is an automorphism is to determine if it is onto. Thus, for example the homomorphism `f1` on the previous page is an automorphism because the image of `f1` is all of S_3. In the following exercises you will need to use the `GroupHomomorphismByImages` command in GAP to find homomorphisms from D_n to D_n. You will then need to check if they are automorphisms by checking to see if the kernel contains only the identity or by checking that the image is all of D_n. Since a homomorphism is completely determined by the image of the generators of a group, you only need to specify where you want to map the two generators of D_n.

The files "autoDn" and "homoDn" contain functions that will list all the automorphisms and homomorphisms of a given dihedral group into itself.

(Thanks to Alexander Hulpke for providing these functions.) Both files are on the web site. The following GAP output is all the automorphisms and then all the homomorphisms of D_6 into itself:

```
gap> Read("autoDn");
gap> Read("homoDn");
gap> d6:= DihedralGroup(IsPermGroup,12);
Group([ (1,2,3,4,5,6), (2,6)(3,5) ])
gap> autoDn(d6);
[ [ (1,2,3,4,5,6), (2,6)(3,5) ] -> [ (1,2,3,4,5,6), (2,6)(3,5) ],
  [ (1,2,3,4,5,6), (2,6)(3,5) ] -> [ (1,2,3,4,5,6), (1,2)(3,6)(4,5) ],
  [ (1,2,3,4,5,6), (2,6)(3,5) ] -> [ (1,2,3,4,5,6), (1,3)(4,6) ],
  [ (1,2,3,4,5,6), (2,6)(3,5) ] -> [ (1,2,3,4,5,6), (1,4)(2,3)(5,6) ],
  [ (1,2,3,4,5,6), (2,6)(3,5) ] -> [ (1,2,3,4,5,6), (1,5)(2,4) ],
  [ (1,2,3,4,5,6), (2,6)(3,5) ] -> [ (1,2,3,4,5,6), (1,6)(2,5)(3,4) ],
  [ (1,2,3,4,5,6), (2,6)(3,5) ] -> [ (1,6,5,4,3,2), (2,6)(3,5) ],
  [ (1,2,3,4,5,6), (2,6)(3,5) ] -> [ (1,6,5,4,3,2), (1,2)(3,6)(4,5) ],
  [ (1,2,3,4,5,6), (2,6)(3,5) ] -> [ (1,6,5,4,3,2), (1,3)(4,6) ],
  [ (1,2,3,4,5,6), (2,6)(3,5) ] -> [ (1,6,5,4,3,2), (1,4)(2,3)(5,6) ],
  [ (1,2,3,4,5,6), (2,6)(3,5) ] -> [ (1,6,5,4,3,2), (1,5)(2,4) ],
  [ (1,2,3,4,5,6), (2,6)(3,5) ] -> [ (1,6,5,4,3,2), (1,6)(2,5)(3,4) ] ]
gap> homoDn(d6);
[ [ (1,2,3,4,5,6), (2,6)(3,5) ] -> [ (), () ],
  [ (1,2,3,4,5,6), (2,6)(3,5) ] -> [ (), (2,6)(3,5) ],
  [ (1,2,3,4,5,6), (2,6)(3,5) ] -> [ (), (1,2)(3,6)(4,5) ],
  [ (1,2,3,4,5,6), (2,6)(3,5) ] -> [ (), (1,3)(4,6) ],
  [ (1,2,3,4,5,6), (2,6)(3,5) ] -> [ (), (1,4)(2,3)(5,6) ],
  [ (1,2,3,4,5,6), (2,6)(3,5) ] -> [ (), (1,4)(2,5)(3,6) ],
  [ (1,2,3,4,5,6), (2,6)(3,5) ] -> [ (), (1,5)(2,4) ],
  [ (1,2,3,4,5,6), (2,6)(3,5) ] -> [ (), (1,6)(2,5)(3,4) ],
  [ (1,2,3,4,5,6), (2,6)(3,5) ] -> [ (2,6)(3,5), () ],
```

(The output continues here listing all 64 homomorphisms.)

```
gap> Size(autoDn(d6));
12
gap> Size(homoDn(d6));
64
```

As a homomorphism is completely determined by its image on a set of generators of the group, GAP only specifies the image of a set of generators of D_6. (Be patient when using these functions; they take awhile.)

Section 7, Project

7.1 **By hand** find three automorphism of D_4. Using GAP determine the number of automorphisms of D_4.

7.2 **By hand** find three homomorphisms from D_4 to D_4 that are not automorphisms. Using GAP determine the number of homomorphisms from D_4 to D_4.

7.3 Repeat Exercises 7.1 and 7.2 for D_5.

7.4 **Using GAP** repeat Exercises 7.1 and 7.2 for D_{19}, D_{21}, D_{45} and D_{49}.

7.5 Make a conjecture about the number of homomorphisms and the number of automorphisms of D_n when n is odd.

7.6 **Using GAP** repeat Exercises 7.1 and 7.2 for D_{20}, D_{24}, D_{48} and D_{50}.

7.7 Make a conjecture about the number of homomorphisms and the number of automorphisms of D_n when n is even.

Below are copies of the files "autoDn" and "homoDn".

The file autoDn:

```
autoDn:= function(G)
local a,b,aims,bims,maps,autos,abims;
a:=G.1; b:=G.2;
aims:=Filtered(Elements(G), i -> Order(a) = Order(i));
bims:=Filtered(Elements(G), i -> Order(b) = Order(i));
abims:= Cartesian(aims,bims);
maps:= List(abims, i -> GroupHomomorphismByImages(G,G,[a,b],i));
maps:= Filtered(maps, i -> i <> fail);
autos:= Filtered(maps, IsInjective);
```

```
return autos;
end;
```

The file homoDn:

```
homoDn:= function(G)
local a,b,aims,bims,maps,homos,abims;
a:=G.1; b:=G.2;
aims:=Filtered(Elements(G), i -> IsInt(Order(a) / Order(i)));
bims:=Filtered(Elements(G), i -> IsInt(Order(b) / Order(i)));
abims:= Cartesian(aims,bims);
maps:= List(abims, i -> GroupHomomorphismByImages(G,G,[a,b],i));
homos:= Filtered(maps, i -> i <> fail);
return homos;
end;
```