Finite Groups In OSCAR

Hal Simpson

Overview

- To create every group up to order 12 in OSCAR, possibly in multiple ways
- To analyse and manipulate these groups using tools provided by OSCAR

The trivial group, to begin with (C_1)

We'll start by creating the cyclic group of order 1 using the cyclic group command.

```
julia> Z_1_1=cyclic_group(1)
Pc group of order 1
```

The trivial group, again

We can also use the abelian group command.

```
julia> X_1_1=abelian_group(1)
Finitely generated abelian group
    with 1 generator and 1 relation and relation matrix
    [1]
```

...Or how about the symmetric group command?

```
julia> S_1_1=symmetric_group(1)
Permutation group of degree 1 and order 1
```

Now with order 2! (C_2)

We can use all the same commands to generate the group of order 2. We can even use the dihedral group command. Note, the dihedral command requires you to input the order of the group, but the symmetric command takes the degree.

```
julia> Z_2_1=cyclic_group(2)
Pc group of order 2

julia> X_2_1=abelian_group(2)
Z/2

julia> S_2_1=symmetric_group(2)
Symmetric group of degree 2

julia> D_2_1=dihedral_group(2)
Pc group of order 2
```

So many choices

These different methods of creating groups are all useful in different ways. There's pros and cons. To demonstrate this, I'll use the non-cyclic group of order 4 as an example.

So many choices ($C_2 \times C_2$)

With the abelian group command, you can make the whole group all in one big go. The symmetric and dihedral commands are both useful for a similar reason.

```
julia> X_4_2=abelian_group(2,2)
(Z/2)^2

julia> D_4_2=dihedral_group(4)
Pc group of order 4
```

So many choices

While working only with cyclic groups means building more complex groups manually, it's the only way to do it in some circumstances. In OSCAR, different group commands produce different types of object that aren't necessarily compatible with each other. Below, I attempt to combine groups created with the "abelian" and "cyclic" commands, and get an error.

```
julia> Z_4_2=direct_product(Z_2_1,X_2_1)
ERROR: [...]

julia> Z_4_2=direct_product(X_2_1,X_2_1)
(Direct product of 2 abelian groups, ...

julia> Z_4_2=direct_product(Z_2_1,Z_2_1)
Direct product of
    pc group of order 2
    pc group of order 2
```

A quick breather

I put this slide here to stop myself from going too fast. Now would be a good time to ask a question, if you have one. If everyone's happy, I'll move on.

Three, four, five (C_3, C_4, C_5)

I'll mostly just be showing the cyclic group commands from hereon, but feel free to try making some of these groups using the other commands.

```
julia> Z_3_1=cyclic_group(3)
Pc group of order 3

julia> Z_4_1=cyclic_group(4)
Pc group of order 4

julia> Z_5_1=cyclic_group(5)
Pc group of order 5
```

Six (C_6)

The cyclic group of order 6 is nothing special. We could even do it as a direct product, if we want to be fancy.

```
julia> Z_6_1=cyclic_group(6)
Pc group of order 6

julia> X_6_1=abelian_group(2,3)
Finitely generated abelian group
  with 2 generators and 2 relations and relation matrix
  [2   0]
  [0   3]
```

The non-abelian group of order 6 is far more interesting.

Six (D_3)

It can be done with the dihedral or symmetric commands, but now would be a great time to figure out how to do semi-direct products in OSCAR.

```
julia> D_6_2=dihedral_group(6)
Pc group of order 6

julia> S_6_2=symmetric_group(3)
Symmetric group of degree 3
```

Semi-direct products are hard

$$G \rtimes_F X$$

As a refresher, a semi-direct product of two groups is their cartesian product along with a homomorphism $F : X \to Aut(G)$, so that every $x \in X$ is mapped onto some $f_x \in Aut(G)$. Multiplication of elements of $G \rtimes_F X$ is as follows:

$$(g, x)(h, y) = (g \cdot f_x(h), x \cdot y)$$

Semi-direct products are hard

The process of making our semi-direct product $C_3 \rtimes_F C_2$ in OSCAR ends with this command.

```
julia> Z_6_2=semidirect_product(Z_3_1,F_6_2,Z_2_1)
Semidirect product of
  normal group: pc group of order 3
  acting group: pc group of order 2
```

Naturally, there's a lot of steps to get to that point.

Semi-direct products are hard

Our action plan is as follows:

- Create the automorphism group of our normal subgroup, C_3;
- Label all necessary automorphisms from our automorphism group;
- Describe our semi-direct product's homomorphism (by using said automorphisms);
- Build the semi-direct product, and we're done!

Semi-direct products are hard

Step 1: Create the automorphism group of C_3.

```
julia> Aut_3_1=automorphism_group(Z_3_1)
Automorphism group of
  pc group of order 3
```

Semi-direct products are hard

Step 2: Wow, that's very complicated looking.

```
julia> f_6_2=Aut_3_1(hom(Z_3_1,Z_3_1,[Z_3_1[0],Z_3_1[1],Z_3_1[-1]],[Z_3_1[0],Z_3_1[-1],Z_3_1[1]]))
[ <identity> of ..., f1, f1^2 ] -> [ <identity> of ..., f1^2, f1 ]
```

Semi-direct products are hard

```
julia> f_6_2=Aut_3_1(hom(Z_3_1,Z_3_1,[Z_3_1[0],Z_3_1[1],Z_3_1[-1]],[Z_3_1[0],Z_3_1[-1],Z_3_1[1]]))
[ <identity> of ..., f1, f1^2 ] -> [ <identity> of ..., f1^2, f1 ]
```

If you were to say it as a proper sentence, you'd say:
"f_6_2 is equal to the element of $Aut(C_3)$ that is the unique homomorphism $C_3 \to C_3$ mapping identity to identity, Z_3_1[1] (the generator of C_3) to its inverse, and vice versa."

Semi-direct products are hard

Actually, you can cut out a lot of redundant info, and just define the homomorphism by mapping generator to generator: "f_6_2 is equal to the element of $Aut(C_3)$ that is the unique homomorphism $C_3 \to C_3$ mapping Z_3_1[1] to its inverse."

```
julia> f_6_2=Aut_3_1(hom(Z_3_1,Z_3_1,[Z_3_1[1]],[Z_3_1[-1]]))
[ f1 ] -> [ f1^2 ]
```

Semi-direct products are hard

Step 3: Defining the homomorphism is comparatively easy, but it's still defining a homomorphism; it's a bit of a faff.

```
julia> F_6_2=hom(Z_2_1,Aut_3_1,[Z_2_1[1]],[f_6_2])
Group homomorphism
  from pc group of order 2
  to automorphism group of pc group of order 3
```

Hopefully you can see what I meant by "define every necessary automorphism": I only needed one automorphism to define this map.

Semi-direct products are hard

Now, let's wrap it up in a nice neat bow.

```
julia> Z_6_2=semidirect_product(Z_3_1,F_6_2,Z_2_1)
Semidirect product of
  normal group: pc group of order 3
  acting group: pc group of order 2
```

Hooray!

Halfway to 12... let's try some tools!

OSCAR has a lot of tools for analysing all the different groups you create. We'll go through some of them here.

Halfway to 12... let's try some tools!

order: Tells you the order of the group.

```
julia> order(Z_4_2)
4
```

Halfway to 12... let's try some tools!

gens: Gives a list of generators for the group (not necessarily minimal).

```
julia> gens(Z_3_1)
1-element Vector{PcGroupElem}:
 f1

julia> gens(Z_4_1)
2-element Vector{PcGroupElem}:
 f1
 f2
```

Halfway to 12... let's try some tools!

elements: Tells you every element of the group, in terms of those generators.

```
julia> elements(Z_5_1)
5-element Vector{PcGroupElem}:
 <identity> of ...
 f1
 f1^2
 f1^3
 f1^4
```

Halfway to 12... let's try some tools!

describe: Describes the group up to isomorphism.

```
julia> describe(S_6_2)
"S3"

julia> describe(X_4_2)
"Z/2 + Z/2"

julia> describe(Z_6_2)
"S3"
```

Halfway to 12... let's try some tools!

is_abelian and is_dihedral_group: Tells you if your group is abelian/dihedral.

```
julia> is_abelian(Z_6_2)
false

julia> is_dihedral_group(Z_6_2)
true
```

Halfway to 12... let's try some tools!

normal_subgroup and acting_subgroup: Tells you what your already know about the semi-direct product you constructed.

```
julia> normal_subgroup(Z_6_2)
Pc group of order 3

julia> acting_subgroup(Z_6_2)
Pc group of order 2
```

Halfway to 12... let's try some tools!

Finally, you can do multiplication of group elements with OSCAR, but you have to use the generating elements from the gens command.

```
julia> gens(Z_6_2)
2-element Vector[...]:
 f1
 f2
julia> Z_6_2[1]
f1

julia> Z_6_2[2]
f2

julia> Z_6_2[1]*Z_6_2[2]*Z_6_2[1]
f2^2
```

That's probably enough for now.

The abelian groups for seven and eight (C_7, C_8, $C_4 \times C_2$)

These ones aren't really special.

```
julia> Z_7_1=cyclic_group(7)
Pc group of order 7

julia> Z_8_1=cyclic_group(8)
Pc group of order 8

julia> Z_8_2=direct_product(Z_4_1,Z_2_1)
Direct product of
  pc group of order 4
  pc group of order 2
```

The abelian groups for seven and eight ($C_2 \times C_2 \times C_2$)

In the case of $C_2 \times C_2 \times C_2$, I'll highlight that you can take the direct product of as many groups as you like.

```
julia> Z_8_3=direct_product(Z_2_1,Z_2_1,Z_2_1)
Direct product of
  pc group of order 2
  pc group of order 2
  pc group of order 2

julia> X_8_3=abelian_group(2,2,2)
(Z/2)^3
```

The dihedral group of order eight (D_4)

Let's get a little bit cute with D_4. Did you know $(C_2 \times C_2) \rtimes C_2$ is isomorphic to D_4?

```
julia> Aut_4_2=automorphism_group(Z_4_2)
Automorphism group of
  Z_2_1 x Z_2_1

julia> gens(Z_4_2)
2-element Vector[...]:
 f1
 f2

julia> Z_4_2[1]
f1

julia> Z_4_2[2]
f2
```

The dihedral group of order eight

I could swap f1 and f2 if I wanted, but that's boring.

```
julia> f_8_4=Aut_4_2(hom(Z_4_2,Z_4_2,[Z_4_2[1],Z_4_2[2]],[Z_4_2[1],Z_4_2[1]*Z_4_2[2]]))
[ f1, f2 ] -> [ f1, f1*f2 ]

julia> F_8_4=hom(Z_2_1,Aut_4_2,[Z_2_1[1]],[f_8_4])
Group homomorphism
  from pc group of order 2
  to automorphism group of Z_2_1 x Z_2_1
```

The dihedral group of order eight

Tada! And just to make sure it worked, I'll check if it's dihedral.

```
julia> Z_8_4=semidirect_product(Z_4_2,F_8_4,Z_2_1)
Semidirect product of
  normal group: Z_2_1 x Z_2_1
  acting group: pc group of order 2

julia> is_dihedral_group(Z_8_4)
true
```

The quaternions (Q_8)

For the quaternion group, there is a command, and it conveniently gives us a group that is compatible with our cyclic groups.

```
julia> Q_8_5=quaternion_group(8)
Pc group of order 8

julia> G=direct_product(Q_8_5,Z_7_1)
Direct product of
  pc group of order 8
  pc group of order 7
```

Nine, ten, eleven... $(C_9, C_3 \times C_3, C_{10}, C_{11})$

Nothing special here.

```
julia> Z_9_1=cyclic_group(9)
Pc group of order 9

julia> Z_9_2=direct_product(Z_3_1,Z_3_1)
Direct product of
  pc group of order 3
  pc group of order 3

julia> Z_10_1=cyclic_group(10)
Pc group of order 10

julia> Z_11_1=cyclic_group(11)
Pc group of order 11
```

EXERCISE: D_5

Take a moment to see if you can construct D_5, using both the dihedral command and the semidirect product command. I've put a reminder of what that looked like for D_3 on the next slide, to help you out.

EXERCISE: D_5

A reminder for how to construct D_3:

```
julia> Aut_3_1=automorphism_group(Z_3_1)
Automorphism group of
  pc group of order 3

julia> f_6_2=Aut_3_1(hom(Z_3_1,Z_3_1,[Z_3_1[1]],[Z_3_1[-1]]))
[ f1 ] -> [ f1^2 ]

julia> F_6_2=hom(Z_2_1,Aut_3_1,[Z_2_1[1]],[f_6_2])
Group homomorphism
  from pc group of order 2
  to automorphism group of pc group of order 3

julia> Z_6_2=semidirect_product(Z_3_1,F_6_2,Z_2_1)
Semidirect product of
  normal group: pc group of order 3
  acting group: pc group of order 2
```

Twelve.

Twelve. (C_{12})

The abelian ones speak for themselves.

```
julia> Z_12_1=cyclic_group(12)
Pc group of order 12
```

Twelve. $(C_6 \times C_2)$

And another.

```
julia> Z_12_2=direct_product(Z_6_1,Z_2_1)
Direct product of
  pc group of order 6
  pc group of order 2
```

Twelve. (D_6)

Let's be cute again, and construct this as the direct product of D_3 and C_2. Indeed, dihedral groups are compatible with cyclic groups in OSCAR.

```
julia> Z_12_3=direct_product(D_6_2,Z_2_1)
Direct product of
  pc group of order 6
  pc group of order 2

julia> is_dihedral_group(Z_12_3)
true
```

Twelve. $(C_3 \rtimes C_4)$

We can reuse our automorphism from constructing D_3 here.

```
julia> f_12_4=f_6_2
[ f1 ] -> [ f1^2 ]

julia> order(Z_4_1[1])
4

julia> F_12_4=hom(Z_4_1,Aut_3_1,[Z_4_1[1]],[f_12_4])
Group homomorphism
  from pc group of order 4
  to automorphism group of pc group of order 3

julia> Z_12_4=semidirect_product(Z_3_1,F_12_4,Z_4_1)
Semidirect product of
  normal group: pc group of order 3
  acting group: pc group of order 4
```

Twelve.

We should probably verify we got the right group. The describe command is rather useful for this.

```
julia> describe(Z_12_4)
"C3 : C4"
```

Twelve. (A_4)

Lastly, there is a special command for A_4.

```
julia> A4=alternating_group(4)
Alternating group of degree 4

julia> order(A4)
12
```

You can generate it as $(C_2 \times C_2) \rtimes C_3$, but I'll leave that as an exercise.

Closing remarks

All these commands and more can be found on the OSCAR website:
https://docs.oscar-system.org/stable/Groups/intro/

If there's extra time after questions, I've got one more group I might like to show you as a bonus.

In any case, thank you for your attention.

Bonus! $(C_7 \rtimes C_3)$

The below group is the smallest non-abelian group of odd order. Like A_4, the acting subgroup has two different non-trivial actions, and that's pretty cool.

```
julia> Aut_7_1=automorphism_group(Z_7_1)
Automorphism group of
  pc group of order 7

julia> f_21=Aut_7_1(hom(Z_7_1,Z_7_1,[Z_7_1[1]],[Z_7_1[1]^2]))
[ f1 ] -> [ f1^2 ]

julia> F_21=hom(Z_3_1,Aut_7_1,[Z_3_1[1]],[f_21])
Group homomorphism
  from pc group of order 3
  to automorphism group of pc group of order 7

julia> G_21=semidirect_product(Z_7_1,F_21,Z_3_1)
Semidirect product of
  normal group: pc group of order 7
  acting group: pc group of order 3

julia> is_abelian(G_21)
false
```