Chapter 1

Library group_example

The following is a commented reading of the formalization of finite groups provided by the MathComp library. The goal is to explain the definitions and the notations, and to replay some of the proofs in a less compact way for didactic purposes. I used the following references for the mathematics [KS04, Ogg11] and received comments from Cyril Cohen.

1.1 Basics About Groups

The formalization of finite groups is essentially built on top of `finset.v`, the formalization of finite sets, hence the following sequence of `Import`s.

In MathComp, finite groups are "subgroups" of a "container group". The first part of the container group structure can be found in the `Record` `mixin_of` (file: `fingroup.v`; `Module` `FinGroup`): a binary operation, a special element (the neutral), a unary operation (inverse), associativity, left identity, involution, antimorphism (i.e., $(xy)^{-1} = y^{-1}x^{-1}$). Such a structure is not a group because $x^{-1}x = 1$ may not hold.

```
Print FinGroup.mixin_of.
```

The carrier of a container group is put together with (1) the assumption that it satisfies the structure above and (2) the assumption that the carrier is a finite type. This forms a `baseFinGroupType`.

```
Print FinGroup.base_type.
Print baseFinGroupType.
```

Then, the `baseFinGroupType` is put together with the following law: $\forall x, x^{-1}x = 1$. This gives a group that will act as a "container group" hereafter.

```
Print FinGroup.type.
Print finGroupType.
```

Finally, a finite group is defined as a set of elements from a finite container group together with the assumption that it is a `group_set`, i.e., that it contains the neutral element and that it is stable by the binary operation. It is worth noticing that the definitive definition of a finite group appears very late in the `fingroup.v` file (almost in the middle of a 3,000 lines file).

```
Print group_set.
Print group_type.
```

How to declare a finite group? First declare a finite container group `gT : finGroupType` and then a finite group using the dedicated notation `G : {group gT}` (scope: `type_scope`, notation for `group_of (Phant gT)`). `{group _}` has type `predArgType` which means that it comes with the generic notation `\in`.

```
Section group_example.

Variable gT : finGroupType.
```

```
Variable G : {group gT}.
```

Groups enjoy the following notations.

```
Local Open Scope group_scope.
```

```
Check (1 : gT).
Check (1 \in G).
Check (1 × 1 : gT).
Check (1 ^-1 : gT).
```

```
Lemma neutral_in_group : 1 \in G.
Check group1.
Qed.
```

```
Lemma neutral_neutral_in_group : 1 × 1 \in G.
Search _ left_id mulg in fingroup.
Qed.
```

Point multiplication and inverse are lifted to set of points.

```
Check set_mulg.
Check set_invg.
```

For two nonempty subsets A, B of G, let $AB := \{ab | a \in A, b \in B\}$. AB is the *(complex) product* of A and B. When $A = \{a\}$, we write aB instead of AB. A product is not necessarily a group (the multiplication needs to commute for that) but it is at least a `group_set_of_baseGroupType`.

```
Check (G × G : group_set_of_baseGroupType gT).
Set Printing All.
Check (1 = [set 1] :> {set gT}).
Unset Printing All.
```

```
Lemma neutral_in_group_group : 1 \in G × G.
Check mulSGid.
Qed.
```

Let A and B be subgroups of G. Then AB is a subgroup of G iff $AB = BA$.

```
Lemma group_set_group_group : group_set (G × G).
Search _ group_set reflect in fingroup.
Search (commute _ _).
Qed.
```

Let U be a subgroup of G and $x \in G$. The product Ux is a *right coset* of U in G. The right coset of H by x is noted `H :* x` in MathComp (notation scope: `group_scope`; file: `fingroup.v`). There is another definition of right cosets (`Definition rcoset`) that is proved equivalent (`Lemma rcosetE`).

```
Variable x : gT.
Variable H : {group gT}.
Check (H :* x : {set gT}).
Locate ":*".
```

The map $H \to Ha; h \mapsto ha$ is injective. Thus a coset Ha has cardinal $|H|$.

```
Lemma mycard_rcoset : #|H :* x| = #| H |.
Check card_rcoset.
Check card_imset.
  Search _ left_injective mulg.
Qed.
```

The set of the right cosets of H by elements of G is denoted by `rcosets H G` (file: `fingroup.v`).

```
Check (rcosets H G : {set {set gT}}).
```

```
Check (rcosets H G).
```

If the set of right cosets of U in G is finite then the number of right cosets of U in G is the *index* of U in G, denoted by $|G : U|$ (`Definition indexg`; file `fingroup.v`).

```
Print indexg.

End group_example.
```

1.2 Lagrange's Theorem

Lagrange's theorem is already proved in `fingroup.v`. In the following, we replay this proof in a less compact way.

```
Check LagrangeI.

Section myLagrange.

Variable gT : finGroupType.

Local Open Scope group_scope.

Variable (H G : {group gT}).
Hypothesis HG : H \subset G.
```

The relation $xy^{-1} \in H$ is an equivalence relation. The equivalence class of x (the set of y such that $xy^{-1} \in H$) is actually the right coset Hx. The set of cosets forms a partition of G. We first prove this fact.

```
Print equivalence_partition.

Lemma rcosets_equivalence_partition :
  rcosets H G = equivalence_partition [rel x y | x × y^-1 \in H] G.

Lemma myrcosets_partition : partition (rcosets H G) G.
```

Lagrange's theorem follows from the fact that the right cosets form a partition of G and that each coset has the same cardinal as H.

```
Lemma myLagrange : #| G | = (#|H| × #|G : H|)%nat.

End myLagrange.
```

1.3 Normal Subgroups

For $x, a \in G$ set $x^a := a^{-1}xa$. This element x^a is the *conjugate* of x by a. (notation: `x ^ y`; notation scope: `group_scope`; file: `fingroup.v`).

```
Print conjg.
Locate "^".
```

Sample property: $x^1 = x$.

```
Check conjg1.
```

For $g \in G$ we set $B^g := g^{-1}Bg$ and say that B^g is the conjugate of B by g. In SSReflect, the conjugate of H by x is denoted by `H :^ x`.

```
Print conjugate.
```

The *normalizer* of A? $\{x | A^x \subseteq A\}$ (Notation: `'N(A)`; definition; file: `fingroup.v`).

```
Print normaliser.
Locate "'N".

Section normalisersect.

Variable gT : finGroupType.
```

```
Variables A B : {group gT}.
Local Open Scope group_scope.
Hypothesis nor : B \subset 'N(A).

Lemma normaliser_equiv b : b \in B → A \subset A :^ b.

End normalisersect.
```

There are many ways to state the fact that a subgroup is normal. For example, a subgroup N of G that satisfies $Nx = xN$ for all $x \in G$ is a *normal* subgroup of G (or is normal in G). We write $N \trianglelefteq G$ if N is normal in G. H is normal in G is noted `H <| G` in MathComp, it is a boolean binary predicate (definition: `normal`; notation scope: `group_scope`; file: `fingroup.v`).

```
Print normal.
```

The following example was originally taken from [BMR+] (`exercises-10.v`).

```
Section normalsect.

Variable gT : finGroupType.
Variables (H G : {group gT}).
Local Open Scope group_scope.
Hypothesis HG : H <| G.

Lemma normal_commutes : H × G = G × H.

End normalsect.
```

Chapter 2

Library permutation_example

```
Set Implicit Arguments.
Import GroupScope.
```

The set of all permutation of `'I_3` forms a `finGroupType`.

```
Goal 'S_3 = {perm 'I_3}.
Check [finGroupType of 'S_3].
```

We define the basic transpositions.

```
Definition p01 : 'S_3 := tperm ord0 (@Ordinal 3 1 erefl).
Definition p02 : 'S_3 := tperm ord0 (@Ordinal 3 2 erefl).
Definition p12 : 'S_3 := tperm (@Ordinal 3 1 erefl) (@Ordinal 3 2 erefl).
```

We define the permutation (021). ($0 \to 2$, $2 \to 1$.)

```
Definition p021 := p01 × p02.
Goal p021 ord0 = @Ordinal 3 1 erefl.
Goal p021 (@Ordinal 3 1 erefl) = @Ordinal 3 2 erefl.
Goal p021 (@Ordinal 3 2 erefl) = ord0.
```

We define the permutation (012).

```
Definition p012 := p02 × p01.
Ltac same_perm :=
  let x := fresh in
  apply/permP ⇒ /= x;
  let x0 := fresh in
  case: (boolP (x == ord0)) ⇒ [/eqP → | x0];
    [by do ! rewrite !permE /= |
     let x1 := fresh in
     case: (boolP (x == @Ordinal 3 1 erefl)) ⇒ [/eqP → | x1];
       [by do ! rewrite !permE /= |
        let x2 := fresh in
        case: (boolP (x == @Ordinal 3 2 erefl)) ⇒ [/eqP → | x2];
          [by do ! rewrite !permE /= |
           (suff : False by done) ;
           case: x x0 x1 x2; case⇒ //; case⇒ //; by case]]].
```

We build the multiplication and inverse tables for permutations of 3 elements.

```
Lemma p01_p01 : p01 × p01 = 1.
Lemma p01_p02 : p01 × p02 = p021.
Lemma p01_p012 : p01 × p012 = p12.
Lemma p01_p12 : p01 × p12 = p012.
Lemma p01_p021 : p01 × p021 = p02.
Lemma p02_p01 : p02 × p01 = p012.
Lemma p02_p12 : p02 × p12 = p021.
Lemma p02_p012 : p02 × p012 = p01.
Lemma p02_p021 : p02 × p021 = p12.
Lemma p12_p01 : p12 × p01 = p021.
Lemma p12_p02 : p12 × p02 = p012.
Lemma p12_p12 : p12 × p12 = 1.
Lemma p12_p012 : p12 × p012 = p02.
Lemma p12_p021 : p12 × p021 = p01.
Lemma p012_p01 : p012 × p01 = p02.
Lemma p012_p02 : p012 × p02 = p12.
Lemma p012_p12 : p012 × p12 = p01.
Lemma p012_p012 : p012 × p012 = p021.
Lemma p012_p022 : p012 × p021 = 1.
Lemma p021_p01 : p021 × p01 = p12.
Lemma p021_p02 : p021 × p02 = p01.
Lemma p021_p12 : p021 × p12 = p02.
Lemma p021_p012 : p021 × p012 = 1.
Lemma p021_p021 : p021 × p021 = p012.
Lemma p01_inv : p01 ^-1 = p01.
Lemma p02_inv : p02 ^-1 = p02.
Lemma p12_inv : p12 ^-1 = p12.
Lemma p012_inv : p012 ^-1 = p021.
Lemma p021_inv : p021 ^-1 = p012.
Lemma p01p12 : p01 != p12.
Lemma p021_1 : p021 ≠ 1.
Lemma p012_1 : p012 ≠ 1.
Lemma p021_neq_p012 : p021 ≠ p012.
Lemma p02_neq_p12 : p02 ≠ p12.
```

We show that S_3 is not commutative.

```
Lemma S_3_not_commutative : ¬ commute p01 p021.
```

A_3 is the group generated by (021), (012), and 1.

```
Definition A_3 : {group 'S_3} := [group of « [set p021; p012; 1] »].
Lemma group_set_A3 : group_set [set p021; p012; 1].
```

Lemma card_A_3 : #| A_3 | = 3.

Lemma card_S_3 : #|[group of « [set x in 'S_3]»]| = 6.

A_3 is normal in S_3.

Lemma A_3_S_3 : A_3 <| [group of « [set x in 'S_3] »].

Bibliography

[BMR+] Yves Berthot, Assia Mahboubi, Laurence Rideau, Pierre-Yves Strub, Enrico Tassi, and Laurent Théry, *International Spring School on Formalization of Mathematics (MAP 2012), March 12–16, 2012, Sophia Antipolis, France*, Available at: http://www-sop.inria.fr/manifestations/MapSpringSchool. Last access: 2014/08/05.

[KS04] Hans Kurzweil and Bernd Stellmacher, *The theory of finite groups—an introduction*, Springer, 2004.

[Ogg11] Frédérique Oggier, *Algebraic methods*, Available at: http://www1.spms.ntu.edu.sg/~frederique/AA11.pdf, Nov. 2011.