A bit of group theory

Enrico Tassi
15 March
Outline

Groups in type theory
  Guided tour to the fingroup library
  Talking about groups in Coq

Lagrange theorem
Why fingroup.v

MathComp Library
7/3/2012
A nonempty set $G$ is a group, if to every $x$ and $y$ both $\in G$ an element $x \ast y \in G$ is assigned, the product of $x$ and $y$, satisfying the following axioms:

**Associativity:** $x \ast (y \ast z) = (x \ast y) \ast z$ for all $x, y, z \in G$

**Existence of identity:** There exists an element $1 \in G$ such that $1 \ast x = x \ast 1 = x$ for all $x \in G$

**Existence of inverses:** For every $x \in G$ there exists an element $x^{-1} \in G$ such that $x \ast x^{-1} = 1 = x^{-1} \ast x$
Looking around

Groups have point-wise operations:

- $g \ast h$, $g^{-1}$
- $g^h := h^{-1} \ast (g \ast h)$
  
  Note: commute $g \ h \rightarrow g^h = g$

- $[\sim g, h] := g^{-1} \ast g^h$
  
  Note: $g \ast h = h \ast g \ast [\sim g, h]$

Groups have mixed operations:

- $H :\ast g := H \ast \{set \ g\}$
- $G :^\sim h := \{set \ g \sim h \mid g \leftarrow G\}$
Looking around

Groups have set-wise operations:

- \( \# |G| \)
- \( G \ast H := \{ \text{set } g \ast h \mid g \leftarrow G, h \leftarrow H \} \)
- \( G :\&: H := \{ \text{set } x \mid (x \in A) \land (x \in B) \} \)
- \( \# |G : H| := \# |\text{rcosets } H G| = \# |\{ \text{set } \text{rcoset } H g \mid g \leftarrow G \}| \)
- \( 'N(G) := \{ \text{set } x \mid G :^* x \subset G \} \)

Groups have predicates:

- \( H <| G := (H \subset G) \land (G \subset 'N(H)) \)

Groups have funny properties:

- \( \text{group_modr } A B G : \)
  \( B \subset G \rightarrow (G :\&: A) \ast B = G :\&: A \ast B \)

Most of the concepts and proofs are at the set level
One may model groups as types:

- very general case (equality means isomorphic), but...
- common stuff is hard to write: \( a \ast c, G :\&: H, \; 'Z(G) \)
- most of the time, \( G \) and \( H \) have a common super group
A global (finite) container equipped with group laws

- a group $A$ is (sub)set that validates

  \[\text{group\_set } A := (1 \in A) \land (A \times A \subseteq A)\]

- constructions are mostly set operations (like $\&\&$, 'Z( ))

- set wise constructions lifted to the group level thanks to Canonical Structure.
**Example**

Lemma `group1` \((gT : \text{finGroupType}) (G : \{\text{group } gT\})\):
\[
1 \in G.
\]

Lemma `example` \((G H : \{\text{group } gT\})\) : \(1 \in G :\&: H\).

```lean
set W := G :\&: H.
```

As you can see, \(G :\&: H\) is a set:

```lean
\[
gT : \text{finGroupType} \\
G : \{\text{group } gT\} \\
H : \{\text{group } gT\} \\
W := G :\&: H : \{\text{set } gT\}
\]
```

But `rewrite group1` lifts it on the fly using a **Canonical Structure** that contains the lemma:

```lean
\[
group_setI (G H : \{\text{group } gT\}) : \text{group_set} (G :\&: H)
\]
```
Finite (intensional) sets

We must find a good “encoding” for sets.

- Sets as characteristic functions
- In Coq functions are not extensional

\[(\forall x. f \ x = g \ x) \not\rightarrow f = g\]

In a finite setting we can represent functions as their graphs, and finite sets as bitmasks

```
1 a b c ... c^-1 b^-1 a^-1
1 0 0 1 ... 0 1 0 0
```

Equal bitmasks, equal sets: \( \forall x. b_1[x] = b_2[x] \) \( \rightarrow b_1 = b_2 \)

\text{setP A B : A =1 B \rightarrow A = B}
Quotients

bad idea

Quotienting $G$ by $H$ makes sense only if $H < | G$:

$$(H :* g_1) :* (H :* g_2) = H :* (g_1 * g_2)$$
Quotients

bad idea

Quotienting $G$ by $H$ makes sense only if $H \triangleleft G$:

$$(H :* g_1) * (H :* g_2) = H :* (g_1 * g_2)$$

$$(H :* g_1) * (H :* g_2) = H :* (g_1 * g_2)$$

$$(H :* g_1) * (H :* g_2) = H :* (g_1 * g_2)$$

$$(H :* g_1) * (H :* g_2) = H :* (g_1 * g_2)$$

$$(H :* g_1) * (H :* g_2) = H :* (g_1 * g_2)$$

$$g_1 *: (H * H) :* g_2 = g_1 *: H :* g_2$$

$$g_1 *: (H * H) :* g_2 = g_1 *: H :* g_2$$

$$g_1 *: (H * H) :* g_2 = g_1 *: H :* g_2$$
Quotients

bad idea

Quotienting $G$ by $H$ makes sense only if $H <| G$:

$$(H :* g_1) * (H :* g_2) = H :* (g_1 * g_2)$$

$$(bad)\hspace{1em}idea:\hspace{1em}G /p H\hspace{1em}where\hspace{1em}p\hspace{1em}proves\hspace{1em}H <| G$$
Quotients

bad idea

Quotienting $G$ by $H$ makes sense only if $H < | G$:

$$(H :* g_1) * (H :* g_2) = H :* (g_1 * g_2)$$

$$\overset{\|}{\|}$$

$$g_1 :* (H * H) :* g_2 = g_1 :* H :* g_2$$

(bad) idea: $G / p H$ where $p$ proves $H < | G$

third_isog : $(G / H / (K / H)) \isog (G / K)$
**Quotients**

**bad idea**

Quotienting $G$ by $H$ makes sense only if $H < | G$:

$$(H :* g1) * (H :* g2) = H :* (g1 * g2)$$

$$g1 *: (H * H) :* g2 = g1 *: H :* g2$$

(bad) idea: $G /p H$ where $p$ proves $H < | G$

third_isog : $(G /p1 H /p2 (K /p3 H)) \isog (G /p4 K)$
Quotients

making \_/ \_ a total operation

(bad) idea: \( G /p H \) where \( p \) proves \( H < | G \)

(good) idea: \( G / H := (G :&: 'N(H)) / H \)
Looking around

Quotients

Lemmas about quotients may or may not require the normality assumption:

▶ quotientMl A B :
   A \subset \mathcal{N}(H) ->
   A \ast B / H = (A / H) \ast (B / H)

▶ quotientI A B :
   (A :&: B) / H \subset A / H :&: B / H
Lemmas about quotients may or may not require the normality assumption:

**Definition** $A \triangleright\triangleright B :=$

if $A :\&: B \subseteq 1\%G$ then ... else set0
Lemmas about quotients may or may not require the normality assumption:

**Definition** \( A \triangleright\!\!\!\!\!\!\!\!\!\downarrow B := \)

\[
\text{if } A :&: B \sub 1\%G \text{ then ... else set0}
\]

**Lemma** `sdprodP` (\( A \ B : \{\text{set gT}\} \) (\( G : \{\text{group gT}\} \)) : 

\( A \triangleright\!\!\!\!\!\!\!\!\!\downarrow B = G \rightarrow \)

\[
[\land \text{are_groups } A \ B, \ A \ast B = G, \ B \sub 'N(A) \land A :&: B = 1]
\]
Outline

Groups in type theory
  Guided tour to the fingroup library
  Talking about groups in Coq

Lagrange theorem
The statement

\[ H \subset G \rightarrow |H| \ast |G : H| = |G| \]
Lemma TutorialLaGrange G H : H \subset G -> (#|H| * #|G : H|)%N = #|G|.
move/setIdPr. move=> {1}<-.
rewrite mulnC. rewrite -sum_nat_const.
rewrite -[#|G|]sum1_card.
rewrite (partition_big_imset (rcoset H)) /= -/(rcosets H G).
apply: eq_bigr => /= Hg.
case/rcosetsP. move=> g Gg -> {Hg}.
rewrite -(card_rcoset _ g).
rewrite -sum1_card.
apply: eq_bigl => k.
rewrite group_modr ?sub1set //.
rewrite inE. congr (_ && _).
rewrite rcosetE. apply/idP/eqP.
  by apply: rcoset_transl.
move=> <-.
rewrite -[X in X \in _](mul1g k).
rewrite mem_mulg //.
by rewrite in_set1.
Qed.
## Counting the teachers

Iterating on the set of teachers

<table>
<thead>
<tr>
<th>Name</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assia</td>
<td>1</td>
</tr>
<tr>
<td>Enrico</td>
<td>1</td>
</tr>
<tr>
<td>Laurent</td>
<td>1</td>
</tr>
<tr>
<td>Laurence</td>
<td>1</td>
</tr>
<tr>
<td>Pierre-Yves</td>
<td>1</td>
</tr>
<tr>
<td>Yves</td>
<td>1</td>
</tr>
</tbody>
</table>

Total: 6
Counting the teachers
Iterating on the set of countries and teachers

1 Assia
1 Laurent
1 Laurence
1 Pierre Yves
1 Yves

5

1 Enrico
1
Lemma partition_big_imset h A F :
\[ \big[\text{aop/idx}\big]_\left(i \in A\right) F i = \big[\text{aop/idx}\big]_\left(j \in h \odot: A\right) \big[\text{aop/idx}\big]_\left(i \in A \mid h i == j\right) F i. \]
The proof
demo

Lemma TutorialLaGrange G H : H \subset G -> (#|H| * #|G : H|)%N = #|G| .
move/setIidPr. move=> {1}<-.
rewrite mulnC. rewrite -sum_nat_const.
rewrite -[#|G|]sum1_card.
rewrite (partition_big_imset (rcoset H)) /= -/(rcosets H G).
apply: eq_bigr => /= Hg.
case/rcosetsP. move=> g Gg -> {Hg}.
rewrite -(card_rcoset _ g).
rewrite -sum1_card.
apply: eq_bigl => k.
rewrite group_modr ?sub1set //.
rewrite inE. congr (_ && _).
rewrite rcosetE. apply/idP/eqP.
    by apply: rcoset_transl.
move=> <-.
rewrite -[X in X \in _](mul1g k).
rewrite mem_mulg //.
by rewrite in_set1.
Qed.