Orbit-Stabiliser Theorem with Application to Rotational Symmetries

Jonas Rädle

December 7, 2022

Abstract

The Orbit-Stabiliser theorem is a simple result in the algebra of groups that factors the order of a group into the sizes of its orbits and stabilisers.

We formalize the notion of a group action and the related concepts of orbits and stabilisers. This allows us to prove the orbit-stabiliser theorem.

In the second part of this work, we formalize the tetrahedral group and use the orbit-stabiliser theorem to prove that there are twelve (orientation-preserving) rotations of the tetrahedron.

Contents

1 Orbit-Stabiliser Theorem **4**
- 1.1 Imports . . . 4
- 1.2 Group Actions . . . 4
- 1.3 Orbit and stabiliser . . . 4
- 1.4 Stabiliser Theorems . . . 5
- 1.5 Picking representatives from cosets . . . 5
- 1.6 Orbit-Stabiliser Theorem . . . 6

2 Rotational Symmetries of the Tetrahedron **6**
- 2.1 Definition of the Tetrahedron and its Rotations . . . 6
- 2.2 Properties of Rotations . . . 7
- 2.3 Inversions . . . 10
- 2.4 The Tetrahedral Group . . . 10
- 2.5 Counting Orbits . . . 11
- 2.6 Counting Stabilisers . . . 11
- 2.7 Proving Finiteness . . . 11
- 2.8 Order of the Group . . . 12

theory *Left-Coset*
imports
HOL−Algebra.Coset

begin

definition
LCOSETS :: [-, *'a set*] ⇒ (*'a set*)*set* (*lcosets*$_1$ - [*81*] *80*)
where *lcosets*$_G$ *H* = (⋃ *a*∈*carrier G*. {*a* <#$_G$ *H*})

definition
LFactGroup :: [(*'a*,*'b*) *monoid-scheme*, *'a set*] ⇒ (*'a set*) *monoid* (**infixl** *LMod 65*)
— Actually defined for groups rather than monoids
where *LFactGroup G H* = (|*carrier* = *lcosets*$_G$ *H*, *mult* = *set-mult G*, *one* = *H*|)

lemma (**in** *group*) *lcos-self*: [| *x* ∈ *carrier G*; *subgroup H G* |] ==> *x* ∈ *x* <# *H*
⟨*proof*⟩

Elements of a left coset are in the carrier

lemma (**in** *subgroup*) *elemlcos-carrier*:
assumes *group G*
assumes *acarr*: *a* ∈ *carrier G*
and *a'*: *a'* ∈ *a* <# *H*
shows *a'* ∈ *carrier G*
⟨*proof*⟩

Step one for lemma *rcos-module*

lemma (**in** *subgroup*) *lcos-module-imp*:
assumes *group G*
assumes *xcarr*: *x* ∈ *carrier G*
and *x'cos*: *x'* ∈ *x* <# *H*
shows (*inv x* ⊗ *x'*) ∈ *H*
⟨*proof*⟩

Left cosets are subsets of the carrier.

lemma (**in** *subgroup*) *lcosets-carrier*:
assumes *group G*
assumes *XH*: *X* ∈ *lcosets H*
shows *X* ⊆ *carrier G*
⟨*proof*⟩

lemma (**in** *group*) *lcosets-part-G*:
assumes *subgroup H G*
shows $\bigcup (lcosets\ H) = carrier\ G$
⟨*proof*⟩

lemma (**in** *group*) *lcosets-subset-PowG*:
$subgroup\ H\ G \implies lcosets\ H \subseteq Pow(carrier\ G)$
⟨*proof*⟩

lemma (**in** *group*) *lcos-disjoint*:
assumes *subgroup H G*
assumes *p*: $a \in lcosets\ H\ b \in lcosets\ H\ a \neq b$
shows $a \cap b = \{\}$
⟨*proof*⟩

The next two lemmas support the proof of *card-cosets-equal*.

lemma (**in** *group*) *inj-on-f'*:
$[\![H \subseteq carrier\ G;\ \ a \in carrier\ G]\!] \implies inj\text{-}on\ (\lambda y.\ y \otimes inv\ a)\ (a <\# H)$
⟨*proof*⟩

lemma (**in** *group*) *inj-on-f''*:
$[\![H \subseteq carrier\ G;\ \ a \in carrier\ G]\!] \implies inj\text{-}on\ (\lambda y.\ inv\ a \otimes y)\ (a <\# H)$
⟨*proof*⟩

lemma (**in** *group*) *inj-on-g'*:
$[\![H \subseteq carrier\ G;\ \ a \in carrier\ G]\!] \implies inj\text{-}on\ (\lambda y.\ a \otimes y)\ H$
⟨*proof*⟩

lemma (**in** *group*) *l-card-cosets-equal*:
$[\![c \in lcosets\ H;\ \ H \subseteq carrier\ G;\ finite(carrier\ G)]\!]$
$\implies card\ H = card\ c$
⟨*proof*⟩

theorem (**in** *group*) *l-lagrange*:
$[\![finite(carrier\ G);\ subgroup\ H\ G]\!]$
$\implies card(lcosets\ H) * card(H) = order(G)$
⟨*proof*⟩

end

1 Orbit-Stabiliser Theorem

In this Theory we will prove the orbit-stabiliser theorem, a basic result in the algebra of groups.

theory *Orbit-Stabiliser*
imports
Left-Coset

begin

1.1 Imports

/HOL/Algebra/Group.thy is used for the definitions of groups and subgroups

Left_Coset.thy is a copy of /HOL/Algebra/Coset.thy that includes additional theorems about left cosets.
The version of Coset.thy in the Isabelle library is missing some theorems about left cosets that are available for right cosets, so these had to be added by simply replacing the definitions of right cosets with those of left cosets.
Coset.thy is used for definitions of group order, quotient groups (operator LMod), and Lagranges theorem.

/HOL/Fun.thy is used for function composition and the identity function.

1.2 Group Actions

We begin by augmenting the existing definition of a group with a group action.
The group action was defined according to [4].

locale *orbit-stabiliser* = *group* +
fixes *action* :: $'a \Rightarrow 'b \Rightarrow 'b$ (**infixl** $\odot$ *51*)
assumes *id-act* [*simp*]: $\mathbf{1} \odot x = x$
and *compat-act*:
$g \in carrier\ G \wedge h \in carrier\ G \longrightarrow g \odot (h \odot x) = (g \otimes h) \odot x$

1.3 Orbit and stabiliser

Next, we define orbit and stabiliser, according to the same Wikipedia article.

context *orbit-stabiliser*
begin

definition *orbit* :: $'b \Rightarrow 'b\ set$ **where**

orbit $x = \{y.\ (\exists\ g \in \mathit{carrier}\ G.\ y = g \odot x)\}$

definition *stabiliser* :: $'b \Rightarrow 'a\ \mathit{set}$
where *stabiliser* $x = \{g \in \mathit{carrier}\ G.\ g \odot x = x\}$

1.4 Stabiliser Theorems

We begin our proofs by showing that the stabiliser forms a subgroup.
This proof follows the template from [2].

theorem *stabiliser-subgroup*: *subgroup* (*stabiliser* x) G
⟨*proof*⟩

As an intermediate step we formulate a lemma about the relationship between the group action and the stabiliser.
This proof follows the template from [3].

corollary *stabiliser-subgroup-corollary*:
assumes *g-car*: $g \in \mathit{carrier}\ G$ **and**
h-car: $h \in \mathit{carrier}\ G$
shows $(g \odot x) = (h \odot x) \longleftrightarrow ((\mathit{inv}\ g) \otimes h) \in \mathit{stabiliser}\ x$
⟨*proof*⟩

Using the previous lemma and our proof that the stabiliser forms a subgroup, we can now show that the elements of the orbit map to left cosets of the stabiliser.
This will later form the basis of showing a bijection between the orbit and those cosets.

lemma *stabiliser-cosets-equivalent*:
assumes *g-car*: $g \in \mathit{carrier}\ G$ **and**
h-car: $h \in \mathit{carrier}\ G$
shows $(g \odot x) = (h \odot x) \longleftrightarrow (g <\# \mathit{stabiliser}\ x) = (h <\# \mathit{stabiliser}\ x)$
⟨*proof*⟩

1.5 Picking representatives from cosets

Before we can prove the bijection, we need a few lemmas about representatives from sets.
First we define rep to be an arbitrary element from a left coset of the stabiliser.

definition *rep* :: $'a\ \mathit{set} \Rightarrow 'a$ **where**
$(H \in \mathit{carrier}\ (G\ \mathit{LMod}\ (\mathit{stabiliser}\ x))) \Longrightarrow \mathit{rep}\ H = (\mathit{SOME}\ y.\ y \in H)$

The next lemma shows that the representative is always an element of its coset.

lemma *quotient-rep-ex* : $H \in (carrier\ (G\ LMod\ (stabiliser\ x))) \Longrightarrow rep\ H \in H$
⟨*proof*⟩

The final lemma about representatives shows that it does not matter which element of the coset is picked, i.e. all representatives are equivalent.

lemma *rep-equivalent*:
  assumes H:$H \in carrier\ (G\ LMod\ stabiliser\ x)$ **and**
    gH:$g \in H$
  shows $H = g <\#\ (stabiliser\ x)$
⟨*proof*⟩

1.6 Orbit-Stabiliser Theorem

We can now establish the bijection between orbit(x) and the quotient group G/(stabiliser(x))

The idea for this bijection is from [1]

theorem *orbit-stabiliser-bij*:
  $bij\text{-}betw\ (\lambda H.\ rep\ H \odot x)\ (carrier\ (G\ LMod\ (stabiliser\ x)))\ (orbit\ x)$
⟨*proof*⟩

The actual orbit-stabiliser theorem is a consequence of the bijection we established in the previous theorem and of Lagrange's theorem

theorem *orbit-stabiliser*:
  assumes *finite*: $finite\ (carrier\ G)$
  shows $order\ G = card\ (orbit\ x) * card\ (stabiliser\ x)$
⟨*proof*⟩
end

end

2 Rotational Symmetries of the Tetrahedron

theory *Tetrahedron*
imports *Orbit-Stabiliser*
begin

2.1 Definition of the Tetrahedron and its Rotations

In this section we will use the orbit-stabiliser theorem to count the number of rotational symmetries of a tetrahedron.

The tetrahedron will be defined as a set of four vertices, labelled A, B, C, and D. A rotation is defined as a function between the vertices.

datatype *Vertex* = *A* | *B* | *C* | *D*
definition *vertices* :: *Vertex set* **where**
vertices = {*A*, *B*, *C*, *D*}

type-synonym *Rotation* = (*Vertex* ⇒ *Vertex*)

We define four primitive rotations explicitly. The axis of each rotation is the line through a vertex that is perpendicular to the face opposite to the vertex. Every rotation is by 120 degrees counter clockwise.
We also define the identity as a possible rotation.

definition *rotate-A* :: *Rotation* **where**
rotate-A = (λ*v*. (*case v of A* ⇒ *A* | *B* ⇒ *C* | *C* ⇒ *D* | *D* ⇒ *B*))
definition *rotate-B* :: *Rotation* **where**
rotate-B = (λ*v*. (*case v of A* ⇒ *D* | *B* ⇒ *B* | *C* ⇒ *A* | *D* ⇒ *C*))
definition *rotate-C* :: *Rotation* **where**
rotate-C = (λ*v*. (*case v of A* ⇒ *B* | *B* ⇒ *D* | *C* ⇒ *C* | *D* ⇒ *A*))
definition *rotate-D* :: *Rotation* **where**
rotate-D = (λ*v*. (*case v of A* ⇒ *C* | *B* ⇒ *A* | *C* ⇒ *B* | *D* ⇒ *D*))

named-theorems *simple-rotations*
declare *rotate-A-def* [*simple-rotations*] *rotate-B-def* [*simple-rotations*] *rotate-C-def* [*simple-rotations*] *rotate-D-def* [*simple-rotations*]

definition *simple-rotations* :: *Rotation set* **where**
simple-rotations = {*id*, *rotate-A*, *rotate-B*, *rotate-C*, *rotate-D*}

All other rotations are combinations of the previously defined simple rotations. We define these inductively.

inductive-set *complex-rotations* :: *Rotation set* **where**
simp: *r* ∈ *simple-rotations* ⟹ *r* ∈ *complex-rotations* |
comp: *r* ∈ *simple-rotations* ⟹ *s* ∈ *complex-rotations* ⟹ (*r* ∘ *s*) ∈ *complex-rotations*

2.2 Properties of Rotations

In this section we prove some basic properties of rotations that will be useful later. We begin by showing that rotations are bijections.

lemma *simple-rotations-inj*:
assumes *r*:*r* ∈ *simple-rotations*
shows *inj r*
⟨*proof*⟩

lemma *simple-rotations-surj*:

assumes r:$r \in$ *simple-rotations*
shows *surj r*
⟨*proof*⟩

lemma *simple-rotations-bij*:
assumes r:$r \in$ *simple-rotations*
shows *bij r*
⟨*proof*⟩

lemma *complex-rotations-bij*: $r \in$ *complex-rotations* $\Longrightarrow$ *bij r*
⟨*proof*⟩

lemma *simple-rotation-bij-corollary*: $r \in$ *simple-rotations* $\Longrightarrow r\ x \neq r\ y \longleftrightarrow x \neq y$
⟨*proof*⟩

lemma *rotation-bij-corollary*: $r \in$ *complex-rotations* $\Longrightarrow r\ x \neq r\ y \longleftrightarrow x \neq y$
⟨*proof*⟩

lemma *complex-rotations-comp*:
$r \in$ *complex-rotations* $\Longrightarrow s \in$ *complex-rotations* $\Longrightarrow (r \circ s) \in$ *complex-rotations*
⟨*proof*⟩

Next, we show that simple rotations (except the identity) keep exactly one vertex fixed.

lemma *simple-rotations-fix*:
assumes r:$r \in$ *simple-rotations*
shows $\exists v.\ r\ v = v$
⟨*proof*⟩

lemma *simple-rotations-fix-unique*:
assumes r:$r \in$ *simple-rotations*
shows $r \neq id \Longrightarrow r\ v = v \Longrightarrow r\ w = w \Longrightarrow v = w$
⟨*proof*⟩

We also show that simple rotations do not contain cycles of length 2.

lemma *simple-rotations-cycle*:
assumes r:$r \in$ *simple-rotations*
shows $r \neq id \Longrightarrow r\ v = w \Longrightarrow v \neq w \Longrightarrow r\ w \neq v$
⟨*proof*⟩

The following lemmas are all variations on the fact that any property that

holds for 4 distinct vertices holds for all vertices. This is necessary to avoid having to use Vertex.exhaust as much as possible.

lemma *distinct-vertices*: $distinct[(a::Vertex),b,c,d] \Longrightarrow (\forall\ e.\ e \in \{a,b,c,d\})$
⟨*proof*⟩

lemma *distinct-map*: $r \in complex\text{-}rotations \Longrightarrow distinct[a,b,c,d] \Longrightarrow (\forall\ e \in \{a,b,c\}.\ r\ e \neq f) \Longrightarrow r\ d = f$
⟨*proof*⟩

lemma *distinct-map'*: $r \in complex\text{-}rotations \Longrightarrow distinct[a,b,c,d] \Longrightarrow (\forall\ e \in \{a,b,c\}.\ r\ f \neq e) \Longrightarrow r\ f = d$
⟨*proof*⟩

lemma *cycle-map*: $r \in complex\text{-}rotations \Longrightarrow distinct[a,b,c,d] \Longrightarrow$
$r\ a = b \Longrightarrow r\ b = a \Longrightarrow r\ c = d \Longrightarrow r\ d = c \Longrightarrow \forall\ v\ w.\ r\ v = w \longrightarrow r\ w = v$
⟨*proof*⟩

lemma *simple-distinct-map*: $r \in simple\text{-}rotations \Longrightarrow distinct[a,b,c,d] \Longrightarrow (\forall\ e \in \{a,b,c\}.\ r\ e \neq f) \Longrightarrow r\ d = f$
⟨*proof*⟩

lemma *simple-distinct-map'*: $r \in simple\text{-}rotations \Longrightarrow distinct[a,b,c,d] \Longrightarrow (\forall\ e \in \{a,b,c\}.\ r\ f \neq e) \Longrightarrow r\ f = d$
⟨*proof*⟩

lemma *simple-distinct-ident*: $r \in simple\text{-}rotations \Longrightarrow distinct[a,b,c,d] \Longrightarrow (\forall\ e \in \{a,b,c\}.\ r\ e \neq e) \Longrightarrow r\ d = d$
⟨*proof*⟩

lemma *id-decomp*:
assumes *distinct*:*distinct* $[(a::Vertex),b,c,d]$ **and** *ident*:$(\forall\ x \in \{a,b,c,d\}.\ r\ x = x)$
shows $r = id$
⟨*proof*⟩

Here we show that two invariants hold for rotations. Firstly, any rotation that does not fix a vertex consists of 2-cycles. Secondly, the only rotation that fixes more than one vertex is the identity.

This proof is very long in part because both invariants have to be proved simultaneously because they depend on each other.

lemma *complex-rotations-invariants*:
$r \in complex\text{-}rotations \Longrightarrow ((\forall\ v.\ r\ v \neq v) \longrightarrow r\ v = w \longrightarrow r\ w = v) \wedge$

$(r\ v = v \longrightarrow r\ w = w \longrightarrow v \neq w \longrightarrow r = id))$
⟨*proof*⟩

This lemma is a simple corollary of the previous result. It is the main result necessary to count stabilisers.

corollary *complex-rotations-fix*: $r \in$ *complex-rotations* $\Longrightarrow r\ a = a \Longrightarrow r\ b = b \Longrightarrow a \neq b \Longrightarrow r = id$
⟨*proof*⟩

2.3 Inversions

In this section we show that inverses exist for each rotation, which we will need to show that the rotations we defined indeed form a group.

lemma *simple-rotations-rotate-id*:
assumes *r*:$r \in$ *simple-rotations*
shows $r \circ r \circ r = id$
⟨*proof*⟩

lemma *simple-rotations-inverses*:
assumes *r*:$r \in$ *simple-rotations*
shows $\exists\, y{\in}$*complex-rotations*. $y \circ r = id$
⟨*proof*⟩

lemma *complex-rotations-inverses*:
$r \in$ *complex-rotations* $\Longrightarrow \exists\, y{\in}$*complex-rotations*. $y \circ r = id$
⟨*proof*⟩

2.4 The Tetrahedral Group

We can now define the group of rotational symmetries of a tetrahedron. Since we modeled rotations as functions, the group operation is functional composition and the identity element of the group is the identity function

definition *tetrahedral-group* :: *Rotation monoid* **where**
tetrahedral-group = (⦇*carrier* = *complex-rotations*, *mult* = (∘), *one* = *id*⦈)

We now prove that this indeed forms a group. Most of the subgoals are trivial, the last goal uses our results from the previous section about inverses.

lemma *is-tetrahedral-group*: *group tetrahedral-group*
⟨*proof*⟩

Having proved that our definition forms a group we can now instantiate our orbit-stabiliser locale. The group action is the application of a rotation.

fun *apply-rotation* :: *Rotation* ⇒ *Vertex* ⇒ *Vertex* **where** *apply-rotation r v* = *r v*

interpretation *tetrahedral*: *orbit-stabiliser tetrahedral-group apply-rotation*
:: *Rotation* ⇒ *Vertex* ⇒ *Vertex*
⟨*proof*⟩

2.5 Counting Orbits

We now prove that there is an orbit for each vertex. That is, the group action is transitive.

lemma *orbit-is-transitive*: *tetrahedral.orbit A* = *vertices*
⟨*proof*⟩

It follows from the previous lemma, that the cardinality of the set of orbits for a particular vertex is 4.

lemma *card-orbit*: *card* (*tetrahedral.orbit A*) = *4*
⟨*proof*⟩

2.6 Counting Stabilisers

Each vertex has three elements in its stabiliser - the identity, a rotation around its axis by 120 degrees, and a rotation around its axis by 240 degrees. We will prove this next.

definition *stabiliser-A* :: *Rotation set* **where**
stabiliser-A = {*id*, *rotate-A*, *rotate-A* ∘ *rotate-A*}

This lemma shows that our conjectured stabiliser is correct.

lemma *is-stabiliser*: *tetrahedral.stabiliser A* = *stabiliser-A*
⟨*proof*⟩

Using the previous result, we can now show that the cardinality of the stabiliser is 3.

lemma *card-stabiliser-help*: *card stabiliser-A* = *3*
⟨*proof*⟩

lemma *card-stabiliser*: *card* (*tetrahedral.stabiliser A*) = *3*
⟨*proof*⟩

2.7 Proving Finiteness

In order to apply the orbit-stabiliser theorem, we need to prove that the set of rotations is finite. We first prove that the set of vertices is finite.

lemma *vertex-set*: (*UNIV*::*Vertex set*) = {*A*, *B*, *C*, *D*}
⟨*proof*⟩

lemma *vertex-finite*: *finite* (*UNIV* :: *Vertex set*)
⟨*proof*⟩

Next we need instantiate Vertex as an element of the type class of finite sets in HOL/Finite_Set.thy. This will allow us to use the lemma that functions between finite sets are finite themselves.

instantiation *Vertex* :: *finite*
begin
instance ⟨*proof*⟩

Now we can show that the set of rotations is finite.

lemma *finite-carrier*: *finite* (*carrier tetrahedral-group*)
⟨*proof*⟩

2.8 Order of the Group

We can now finally apply the orbit-stabiliser theorem. Since we have orbits of cardinality 4 and stabilisers of cardinality 3, the order of the tetrahedral group, and with it the number of rotational symmetries of the tetrahedron, is 12.

theorem *order tetrahedral-group* = *12*
⟨*proof*⟩

end

end

References

[1] Proofwiki. Orbit-stabilizer theorem. https://proofwiki.org/wiki/Orbit-Stabilizer_Theorem, 2017. [Online; accessed 18-July-2017].

[2] Proofwiki. Stabilizer is subgroup. https://proofwiki.org/wiki/Stabilizer_is_Subgroup, 2017. [Online; accessed 18-July-2017].

[3] Proofwiki. Stabilizer is subgroup corollary 2. https://proofwiki.org/wiki/Stabilizer_is_Subgroup/Corollary_2, 2017. [Online; accessed 18-July-2017].

[4] Wikipedia. Group action. https://en.wikipedia.org/wiki/Group_action, 2017. [Online; accessed 18-July-2017].